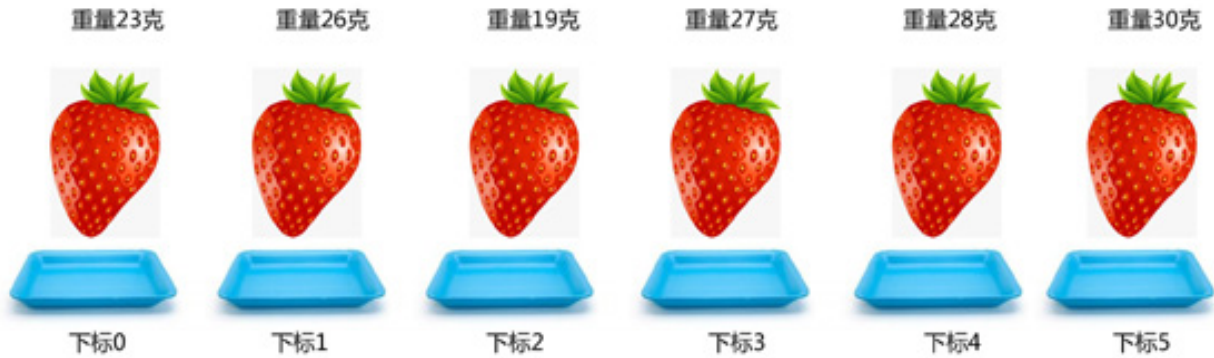


数组

一维数组基础



4.0-1

```
#include<iostream>
using namespace std;
int main()
{
    int a[6]={23, 26, 19, 27, 28, 30};
    cout<<a[5];
    return 0;
}
```

数组就是一批数据，用数组来解决数据的批量存储、计算等问题。

下标：数据存放位置

a[6]

a 数组名（名字为 a 的数组）

6 这组数据只有 6 个，下标从 0 至 5

cout<<a[5]; 修改数字 5, 观察输出变化, 观察下标对应的草莓。

4.0-2

```
#include<iostream>
using namespace std;
int main()
{
    int i;
    int a[6]={23, 26, 19, 27, 28, 30};
    for (i=0; i<=5; i++)
        cout<<a[i]<<" ";
    return 0;
}
```

正序输出

i=0; 从下标 0 开始

i<=5; 到下标 5 结束

i++ 下标每次递增 1

4.0-3

观察输出变化

```
#include<iostream>
using namespace std;
int main()
{
    int i;
    int a[6]={23, 26, 19, 27, 28, 30};
    for (i=1; i<=3; i++)
        cout<<a[i]<<" ";
    return 0;
}
```

4.0-4

$i+=2$ 就是 $i=i+2$, 每次递增 2

```
#include<iostream>
using namespace std;
int main()
{
    int i;
    int a[6]={23, 26, 19, 27, 28, 30};
    for (i=0; i<=5; i+=2)
        cout<<a[i]<<" ";
    return 0;
}
```

观察输出变化

4.0-5

倒序输出

```
#include<iostream>
using namespace std;
int main()
{
    int i;
    int a[6]={23, 26, 19, 27, 28, 30};
    for (i=5; i>=0; i--)
        cout<<a[i]<<" ";
    return 0;
}
```

$i--$ 每次递减 1

4.0-6

正序输入

```
#include<iostream>
```

```
using namespace std;
```

正序输出

```
int main()
```

```
{
```

```
    int a[6];
```

```
    int i;
```

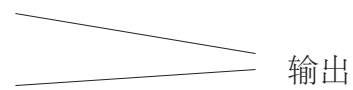
```
    for (i=0; i<=5; i++)
```

```
        cin>>a[i];
```



```
    for (i=0; i<=5; i++)
```

```
        cout<<a[i]<<" ";
```



```
    return 0;
```

```
}
```

4.0-7

正序输入

```
#include<iostream>
```

```
using namespace std;
```

逆序输出

```
int main()
```

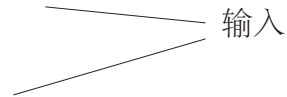
```
{
```

```
    int a[6];
```

```
    int i;
```

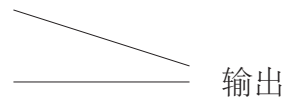
```
    for (i=0; i<=5; i++)
```

```
        cin>>a[i];
```



```
    for (i=5; i>=0; i--)
```

```
        cout<<a[i]<<" ";
```



```
    return 0;
```

```
}
```

4.0-8

```
#include<iostream>// 几个数字相加
using namespace std;
int main()
{
    int i, n, jiafa;//chengfa 加法
    int a[10000];// 预设一个能保存 1 万个数据的数组
    jiafa=0;// 几个数字的和, 最初的数值
    cin>>n;// 输入数字的个数
    for (i=1;i<=n;i++) cin>>a[i];// 输入数组
    for (i=1;i<=n;i++)// 几个数字相乘
    {
        jiafa=jiafa+a[i];
    }
    cout<<jiafa;
    return 0;
}
```

```
3
2 3 4
9
```

尝试这样写:

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

4.0-9

```
#include<iostream>// 几个数字相乘
using namespace std;
int main()
{
    int i, n, chengfa;//chengfa 乘法
    int a[10000];// 预设一个能保存 1 万个数据的数组
    chengfa=1;// 乘积最初的数值 (为什么 jiafa=0, chengfa=1?)
    cin>>n;// 输入数字的个数
    for (i=0;i<n;i++) cin>>a[i];// 输入数组, 如果写成 i=1;i<=n 作用一样

    for (i=0;i<n;i++)// 几个数字相乘
    {
        chengfa=chengfa*a[i];
    }
    cout<<chengfa;
    return 0;
}
```

```
3
2 3 4
24
```

4.0-10

输入 n 个数字，输出 3 的倍数

```
#include<iostream>// 输出 3 的倍数
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a[10000];
```

```
    int i, n, s=0;
```

```
    cin>>n;
```

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

```
    for (i=0; i<n; i++)
```

```
    {
```

```
        if(a[i]%3==0)
```

```
        {
```

```
            cout<<a[i]<<" ";
```

```
            s++;
```

```
        }
```

```
    }
```

```
    cout<<endl<<s;
```

```
    return 0;
```

```
}
```

```
5
33 66 87 67 34
33 66 87
3
```

练习题:

1、输入 n 个数字，输出偶数

2、输入 n 个数字，输出奇数

3、输入 n 个数字，输出 3、5 公倍数

30 50 60 90 78 66

4、输入 n 个数字，输出 20、30 公约数

2 5 3 14 25 6

5、输入 n 个数字，输出质数

7 3 14 26 78

6、输入 n 个数字，求其平均值（准确到小数点后三位数）

练习题:

1、统计班里 n 名同学成绩，算出总分，平均分。

输入: 3

输出: 211

输入: 5

输出: 453

90 70 51

70.3

90 90 90 90 93

90.6

2、商店里一共 x 元商品，卖出 y 件，求总共卖出多少元商品，平均价格是多少，剩余商品价值。

输入: 100

(商品总价格)

输出: 14

3

(商品出售数量)

4.7 (四舍五入保留 1 位)

4.6 4.4 5 (单价)

85.3 (四舍五入保留 1 位)

输入: 200

输出: 51

5

10.2

10 20 10 5 6

149.00

4.0-11// 本题选自《CCF 中学生计算机程序设计》

```
#include<iostream>// 输出最大数字的位置 (下标)
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a[10000];
```

```
    int i, n, max, k;//k 用来记录最大数的下标
```

```
    cin>>n;
```

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

```
    max=a[1];// 暂时假设第一个数字是最大数
```

```
    k=1;// 哪个是最大数, k 就等于哪个数字的下标
```

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

```
    {
```

```
        if(a[i]>max) {    max=a[i];k=i;    }
```

```
    }
```

```
    cout<<k;
```

```
    return 0;
```

```
}
```

4.0-12// 本题选自《CCF 中学生计算机程序设计》

```
#include<iostream>// 斐波那契数列
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    long long a[10000];
```

```
    int i, n;
```

```
    a[0]=0;
```

```
    a[1]=1;
```

```
    cin>>n;
```

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

```
    {
```

```
        a[i]=a[i-2]+a[i-1];
```

```
    }
```

```
    for (i=0;i<n;i++) cout<<a[i]<<" ";
```

```
    return 0;
```

```
}
```

```
5
```

```
45 36 77 23 15
```

```
3
```

练习题:

输出最大数、最小数,
及其下标。

输入 n, 输出 n 个数字

```
20
```

```
0 1 1 2 3 5 8 13 21
```

```
34 55 89 144 233 377
```

```
610 987 1597 2584
```

```
4181
```

练习题:

求前 10 位数字之和,
平均值。

增加、删除、插入、移动

4.1-1

```
#include<iostream> // 删除数据
using namespace std;
int main()
{
    int n, i, x, q[10000];
    cin>>n; // 输入数字的数量 (几个数?)
    for (i=1; i<=n; i++) cin>>q[i]; // 输入数组
    cin>>x; // 输入删除数字的位置 (x 是下标)

    for (i=x; i<n; i++) q[i]=q[i+1]; // 从下标 x 至下标 n-1, 数据向前挪一位

    for (i=1; i<n; i++) cout<<q[i]<<" ";
    return 0;
}
```

```
5
2 5 7 9 8
3
2 5 9 8
```

4.1-2

```
#include<iostream> // 增加数据
using namespace std;
int main()
{
    int n, i, x, q[10000];
    cin>>n; // 输入数字的数量 (几个数?)
    for (i=1; i<=n; i++) cin>>q[i]; // 输入数组
    cin>>x; // 输入增加数字的位置 (x 是下标)

    for (i=n; i>=x; i--) q[i+1]=q[i]; // 从下标 n 至下标 x, 数据向后挪一位

    cin>>q[x]; // 增加的数字
    for (i=1; i<=n+1; i++) cout<<q[i]<<" ";
    return 0;
}
```

```
5
2 5 7 9 8
3
16
2 5 16 7 9 8
```


4.1-3

`#include<iostream>`// 插入删除数据, 最后一个数字插在 x 下标位置

`using namespace std;`

`int main()`

`{`

`int n, i, x, q[10000];`

`cin>>n;`// 输入数字的数量 (几个数?)

`for (i=1; i<=n; i++) cin>>q[i];`// 输入数组

`cin>>x;`// 输入插入数字的位置 (x 是下标)

`for (i=n; i>=x; i--) q[i+1]=q[i];`// 从下标 n 至下标 x, 数据向后挪一位

`q[x]=q[n+1];`// 数组最后一个数字放在下标 x 这个位置

`for (i=1; i<=n; i++) cout<<q[i]<<" ";`

`return 0;`

`}`

```
5
2 5 7 9 8
3
2 5 8 7 9
```

4.1-4

`#include<iostream>`// 数字前移一个位置

`using namespace std;`

`int a[10], i;`

`int main()`

`{`

`for (int i=0; i<10; i++)`

`cin>>a[i];`

`int temp=a[0];`

`for (i=0; i<10-1; i++)`

`a[i]=a[i+1];`

`a[10-1]=temp;`

`for (i=0; i<10; i++)`

`cout<<a[i]<<" ";`

`return 0;`

`}`

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

线性查找

4.2-1

```
#include<iostream>// 线性查找
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, i, num, g[10000];
```

```
    cin>>n;// 输入数字数量
```

```
    for (i=1; i<=n; i++) cin>>g[i];// 输入数组
```

```
    cin>>num;// 输入要查找的数字
```

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

```
    {
```

```
        if (g[i]==num)// 如果哪个数据等于这个数字
```

```
        {
```

```
            cout<<i<<" ";// 则输出这个数据的下标
```

```
        }
```

```
    }
```

```
    return 0;
```

```
}
```

```
5
2 5 7 9 8
5
2
```

```
5
2 5 7 5 8
5
2 4
```

4.2-2

```
#include<iostream>// 线性查找
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, i, num, jiang, haoma, g[10000];
```

```
    cin>>n;// 输入数字数量
```

```
    for (i=1;i<=n;i++) cin>>g[i];// 输入数组
```

```
    cin>>num;// 输入要查找的数字
```

```
    jiang=0;// 预设中奖为 0
```

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

```
    {
```

```
        if(g[i]==num)// 如果查到该数字
```

```
        {
```

```
            jiang=1;// 奖等于 1
```

```
            haoma=i;// 号码等于数据在数组中的位置
```

```
        }
```

```
    }
```

```
    if(jiang==0)cout<<jiang;// 如果奖为 0, 就是中奖数字没有, 输出 0
```

```
    else      cout<<haoma;// 如果奖为 1, 输出数字在数组中的下标
```

```
    return 0;
```

```
}
```

```
6
16 25 37 49 55 21
88
0
```

```
6
16 25 37 49 55 21
37
3
```

二分查找

4.2-3

```
#include<iostream>// 二分查找，必须是有序排列好的数组
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int i, n, w, f, left, right, mid, g[100]; //left 左, right 右, mid 中间
```

```
    cin>>n; // 输入数字数量 (几个数?)
```

```
    for (i=1; i<=n; i++) cin>>g[i];
```

```
    cin>>w; //w 查找的数
```

```
    //////////////////////////////////////// 二分查找程序
```

```
    f=0;
```

```
    left=1; right=n;
```

```
    while (left<=right)
```

```
    {
```

```
        mid=(left+right)/2;
```

```
        if (w==g[mid])    {    f=mid;    break;}
```

```
        if (w<g[mid])    {    right=mid-1;    }
```

```
        if (w>g[mid])    {    left=mid+1;    }
```

```
    }
```

```
    cout<<f;
```

```
    return 0;
```

```
}
```

```
6
5 6 7 8 9 10
6
2
```

布尔变量

bool 布尔变量 表示相反状态

真或者假，成立或者不成立，符合条件或者不符合条件，是或者不是，有或者没有……

布尔 (boolean) 型变量只能取两个值，True 和 False。当把布尔型变量用在数字运算中，用 1 和 0 代表 True 和 False。

非 0 为真，0 为假。

一本通训练指导 6、8、16 是布尔变量题。

4.3-0

```
#include<iostream>// 校门外的树
using namespace std;
int main()
{
    bool a[10001];
    int m, l, i, j, x, y, s=0;
    //l 区域总长度, m 要挪走树的区域
    cin>>l>>m;
    //初始化为树木都存在, L+1 棵树
    for (i=0; i<=l; i++)    a[i]=1;
    //m 个区域的树被挪走
    for (i=1; i<=m; i++)
    {
        cin>>x>>y;
        for (j=x; j<=y; j++) a[j]=0;
    }
    for (i=0; i<=l; i++)
    {
        if(a[i]==1)
            s++;
    }
    cout<<s;
    return 0;
}
```

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。一些区域的树要挪走，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。这些区域中的树（包括区域端点处的两棵树）移走。计算将这些树都移走后，马路上还有多少棵树。

输入：第一行有两个整数 L ($1 \leq L \leq 10000$) 和 M ($1 \leq M \leq 100$)，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：包括一行，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：	样例输出：
500 3	298
150 300	
100 200	
470 471	

题目描述：一个数字，只含 7 或者 4，是幸运数字。

4.3-1-a

```
#include<iostream>// 幸运数
using namespace std;// 本题选自《课课通》
int main()
{
    int num;
    bool flag=true;// 标志，假设都是幸运数
    cin>>num;
    while(num!=0)
    {
        cout<<num%10<<" ";
        if(num%10!=4&&num%10!=7)
            flag=false;

        num=num/10;
    }
    if(flag==true)
        cout<<endl<<" 是幸运数 ";
    else
        cout<<endl<<" 不是幸运数 ";

    return 0;
}
```

```
744
4 4 7
是幸运数
```

4.3-1-b

```
#include<iostream>// 输出幸运数字下标
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, i, num[10000];
```

```
    bool flag;// 标志, 假设都是幸运数
```

```
    cin>>n;
```

```
    for (i=1;i<=n;i++) cin>>num[i];
```

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

```
    {
```

```
        flag=true;
```

```
        while (num[i]!=0)
```

```
        {
```

```
            if (num[i]%10!=4&&num[i]%10!=7)
```

```
                flag=false;
```

```
            num[i]=num[i]/10;
```

```
        }
```

```
        if (flag==true)
```

```
            cout<<" 第 "<<i<<" 个数字 "<<" 是幸运数 "<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

5

56 47 774 15 16

第 2 个数字是幸运数

第 3 个数字是幸运数

4.3-1-c

```
#include<iostream>// 输出幸运数字
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, i;
```

```
    bool flag; // 假设都是幸运数
```

```
    int num[1000], num2[1000];
```

```
    cin>>n;
```

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

```
    {
```

```
        cin>>num[i];
```

```
        num2[i]=num[i];
```

```
    }
```

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

```
    {
```

```
        flag=true; // 假设都是幸运数
```

```
        while (num[i]!=0)
```

```
        {
```

```
            if (num[i]%10!=4&&num[i]%10!=7)
```

```
                flag=false;
```

```
            num[i]=num[i]/10;
```

```
        }
```

```
        if (flag==true)
```

```
            cout<<num2[i]<<" ";
```

```
    }
```

```
    return 0;
```

```
}
```

```
5  
12 16 47 774 777  
47 774 777
```


4.3-1-d

```
#include<iostream>// 幸运数放在一个数组
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, i, j=0;
```

```
    bool flag; // 假设都是幸运数
```

```
    int num[1000], num2[1000], num3[1000];
```

```
    cin>>n;
```

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

```
    {
```

```
        cin>>num[i];
```

```
        num2[i]=num[i];
```

```
    }
```

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

```
    {
```

```
        flag=true; // 假设都是幸运数
```

```
        while (num[i]!=0)
```

```
        {
```

```
            if (num[i]%10!=4&&num[i]%10!=7) flag=false;
```

```
            num[i]=num[i]/10;
```

```
        }
```

```
        if (flag==true)
```

```
        {
```

```
            j=j+1;
```

```
            num3[j]=num2[i];
```

```
        }
```

```
    }
```

```
    cout<<endl;
```

```
    for (i=1; i<=j; i++)
```

```
    {
```

```
        cout<<num3[i]<<" ";
```

```
    }
```

```
    return 0;
```

```
}
```

```
5  
56 47 74 774 35  
  
47 74 774
```

4.3-2-a

```
#include<iostream>// 站着的人和坐下的人
using namespace std;
int main()// 本题选自《CCF 中学生计算机程序设计》5.9
{
    bool a[10000];
    int n,k,i,j;
    cin>>n>>k;//n个人,操作到k人及k的倍数
    for(i=1;i<=n;i++) a[i]=true;//初始化为真,表示站着

    cout<<endl<<"初始化结果:"<<endl;
    for(i=1;i<=n;i++) cout<<a[i]<<" ";//观察初始化结果
    //////////////////////////////////////
    for(i=2;i<=k;i++)//i增大到k为止
    {
        for(j=1;j<=n;j++)//n个人挨个操作
        {
            if(j%i==0)
                a[j]=!a[j];//j是i的倍数,执行相反操作
        }
    }
    //////////////////////////////////////
    cout<<endl<<"所有的人:";
    for(i=1;i<=n;i++)
        cout<<a[i]<<" ";//观察所有人

    cout<<endl<<"站着的人:";
    for(i=1;i<=n;i++)
    {
        if(a[i]==true)
            cout<<i<<" ";
    }
    return 0;
}
```

7 3

初始化结果:

1 1 1 1 1 1 1

所有的人: 1 0 0 0 1 1 1

站着的人: 1 5 6 7

4.3-2-b

```
#include<iostream>// 站着的人和坐下的人
using namespace std;
int main()
{
    bool a[10000];
    int n,k,i,j;
    cin>>n>>k;//n个人,操作到k人及k的倍数
    for(i=1;i<=n;i++) a[i]=1;//初始化为1,表示站着

    cout<<endl<<"初始化结果:";
    for(i=1;i<=n;i++) cout<<a[i]<<" ";//观察初始化结果
    //////////////////////////////////////
    for(i=2;i<=k;i++)//i增大到k为止
    {
        for(j=1;j<=n;j++)//n个人挨个操作
        {
            if(j%i==0)
                a[j]=!a[j];//j是i的倍数,执行相反操作
        }
    }
    //////////////////////////////////////
    cout<<endl<<"所有的人:";
    for(i=1;i<=n;i++)
        cout<<a[i]<<" ";//观察所有人

    cout<<endl<<"站着的人:";
    for(i=1;i<=n;i++)
    {
        if(a[i]==1)
            cout<<i<<" ";
    }
    return 0;
}
```

7 3

初始化结果: 1 1 1 1 1 1 1
所有的人: 1 0 0 0 1 1 1
站着的人: 1 5 6 7

4.3-2-c

```
#include<iostream>// 站着的人和坐下的人
#include<cstring>
using namespace std;
int main()
{
    int a[10000];
    int n,k,i,j;
    memset(a,0,sizeof(a));// 初始化为 0, 表示站着
    cin>>n>>k;//n 个人, 操作到 k 人及 k 的倍数
    for (i=2;i<=k;i++)//i 增大到 k 为止
    {
        for (j=1;j<=n;j++)//n 个人挨个操作
        {
            if(j%i==0)
                a[j]=!a[j];//j 是 i 的倍数, 执行相反操作
        }
    }
    for (i=1;i<=n;i++)
    {
        if(a[i]==0)
            cout<<i<<" ";
    }
    return 0;
}
```

```
7 3
1 5 6 7
```

4.3-3

`#include<iostream>`// 房间开关，最初都打开（开灯关灯）

`#include<cstring>`// 本题选自《一本通》

`#define maxn 100+10`// 数组设置大一些

`using namespace std;`

`int a[maxn];`

`int main()`

`{`

`int i, j, n;`

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

`for (i=1; i<=100; i++)`//100 个服务员操作

`{`

`for (j=1; j<=100; j++)`//100 个房间

`{`

`if (j%i==0) a[j]=!a[j];`// 如果可以整除，相反操作

`}`

`}`

`for (i=1; i<=100; i++)`

`{`

`if (a[i])`// 如果 a[i] 为 1

`{`

`cout<<i<<" ";`

`}`

`}`

`return 0;`

`}`

1 4 9 16 25 36 49 64 81 100

4.3-4

```
#include<iostream>// 筛选法求素数
#include<cmath>// 数学函数库
using namespace std;
int main()//1 不是素数, 也不是合数, 2 是素数
{
    int n, i, j;
    bool p[100001];
    for (i=0; i<=100001; i++) p[i]=true;// 假设都是素数

    cin>>n;// 求 2-n 之间的素数
    //////////////////////////////////////// 筛选法核心程序

    for (i=2; i<=sqrt(n); i++) //sqrt 求平方根函数, i<n 也可以, 但是效率低
// 从 2 开始, 到 n 的平方根结束, 把这些数字的倍数都设成 false, 即不是素数
    {
        if(p[i])//p[i]==true 的简省写法
        {
            for (j=2; i*j<=n; j++) p[i*j]=false;//i 的倍数设成 false
        }
    }

    //////////////////////////////////////// 输出素数
    for (i=2; i<=n; i++)
    {
        if(p[i]) cout<<" "<<i;
    }
    return 0;
}
```

```
100
2 3 5 7 11 13 17 19 23 29 31
37 41 43 47 53 59 61 67 71
73 79 83 89 97
```

筛选法求素数, 以求 100 之内素数为例讲解:

2 的倍数不是素数, 3 的倍数不是素数……一直到 10 的倍数不是素数。

10 是 100 的平方根, 没必要再继续测试了, 从 11 到 99 数字的倍数, 也是从 2 到 10 的某个数字的倍数。

4.3-5// 本题选自《CCF 中学生计算机程序设计》

```
#include<iostream>// 约瑟夫问题
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int n, m, i, s1, s2;//n 只猴子数到 m 出圈
```

```
    bool a[10000];// 用 int 也可以
```

```
    cin>>n>>m;
```

```
    for (i=1;i<=n;i++)    a[i]=1;// 每只猴子都标记 1, 表示这只猴子在圈内
```

```
    s2=n;// 标记剩下的猴子
```

```
    s1=0;//s1 记录从 1 数到 m 只猴子
```

```
////////////////////////////////////
```

```
    while (s2>1)// 剩余最后一只猴子停止
```

```
    {
```

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

```
        {    // 如果猴子数据为 1, s1 加 1 ; 如果猴子数据为 0, 跳过去, 不统计
```

```
            if(a[i]==1) s1++;
```

```
            if(s1==m)
```

```
            {
```

```
                a[i]=0;// 如果数到 M 只猴子, 第 m 只猴变成 0;
```

```
                cout<<i<<" ";// 输出出圈的猴子
```

```
                s1=0;//s1 归 0, 从头开始统计;
```

```
                s2--;// 剩余猴子减 1
```

```
            }
```

```
        }
```

```
    }
```

```
////////////////////////////////////
```

```
    for (i=1;i<=n;i++)// 从头数数, 哪个不是 0, 输出下标
```

```
    {
```

```
        if(a[i]!=0)
```

```
            cout<<endl<<i;
```

```
    }
```

```
    return 0;
```

```
}
```

```
8 5
5 2 8 7 1 4 6
3
```

一维数组练习题

4.5-1// 本题选自《信息学奥赛课课通》

`#include<iostream>`// 寻找比自己学习好的，而且成绩最接近自己

`using namespace std;`

`int main()`

`{`

`int n, i, j, ans, max, h[1000];`

`cin>>n;`

`for (i=1; i<=n; i++) cin>>h[i];`

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

`{`

`ans=0;`//ans 答案，暂时预设为 0

`max=100000;`// 暂时设置一个很高的学习成绩

`for (j=1; j<=n; j++)`

`{`

`if (h[j]>h[i]&&h[j]<max)`

`{`

`ans=j;`

`max=h[j];`

`}`

`}`

`cout<<ans<<" ";`

`}`

`return 0;`

`}`

```
6
3 2 6 1 1 2
3 1 0 2 2 1
```


4.5-2// 本题选自《信息学奥赛课课通》

```
#include<iostream>// 比身高
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int h[10000];
```

```
    int n, i, j, t1, t2, ans=0;
```

```
    cin>>n;
```

```
    for (i=1;i<=n;i++) cin>>h[i];
```

```
    //////////////////////////////////////
```

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

```
    {
```

```
        t1=t2=0;
```

```
        for (j=1;j<i;j++)
```

```
        {
```

```
            if(h[j]>h[i]) t1++;// 排在他前面比他高的人数
```

```
        }
```

```
        for (j=i+1;j<=n;j++)
```

```
        {
```

```
            if(h[j]>h[i]) t2++;// 排在他后面比他高的人数
```

```
        }
```

```
        if(t1==t2)
```

```
        {
```

```
            cout<<i<<" ";// 输出符合条件的人的位置(下标)
```

```
            ans++;// 符合条件的人增加一个
```

```
        }
```

```
    }
```

```
    cout<<" 满足条件的人数: "<<ans;
```

```
    return 0;
```

```
}
```

【题目描述】有 N 个人排成一排，假设他们的身高均为正整数，请找出其中符合以下条件的人：排在他前面且比他高的人数与排在他后面且比他高的人数相等。

【输入】第一行为一个正整数 N ， $1 < N < 1000$ ，表示有多少个人。

下面 N 行，每行一个正整数，表示从前往后每个人的身高，假设每个人的身高 ≤ 10000 。

【输出】一行一个整数，表示满足这个条件的人数。

【输入样例】4 1 2 1 3

【输出样例】2

【样例说明】第 3、第 4 个人满足条件。

```
4
1 2 1 3
3 4 满足条件的人数:
2
```

4.5-3-a// 本题选自《CCF 中学生计算机程序设计》

```
#include<iostream>// 最大连续段和
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a[10000];
```

```
    int n, i, j, k;
```

```
    int max;// 子段和最大值
```

```
    int temp;// 子段和
```

```
    cin>>n;
```

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

```
    max=a[1];
```

```
    for (i=1; i<=n; i++)// 子段起始位置
```

```
    {
```

```
        for (j=i; j<=n; j++)// 子段终点位置
```

```
        {
```

```
            temp=0;
```

```
            for (k=i; k<=j; k++)
```

```
            {
```

```
                temp=temp+a[k];
```

```
                cout<<a[k]<<" ";// 观察程序运行
```

```
            }
```

```
            cout<<endl;// 分行观察程序运行
```

```
            if (max<temp) max=temp;
```

```
        }
```

```
        cout<<"*****"<<endl;// 观察程序运行
```

```
    }
```

```
    cout<<" 最大子段和: " <<max;
```

```
    return 0;
```

```
}
```

```
5
1 -2 3 1 -4
1
1 -2
1 -2 3
1 -2 3 1
1 -2 3 1 -4
*****
-2
-2 3
-2 3 1
-2 3 1 -4
*****
3
3 1
3 1 -4
*****
1
1 -4
*****
-4
*****
最大子段和: 4
```

4.5-3-b// 本题选自《CCF 中学生计算机程序设计》

```
#include<iostream>// 最大连续段和
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a[10000];
```

```
    int n, i, j, k;
```

```
    int max;// 子段和最大值
```

```
    int temp;// 子段和
```

```
    cin>>n;
```

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

```
    max=a[1];
```

```
    for (i=1;i<=n;i++)// 子段起始位置
```

```
    {
```

```
        for (j=i;j<=n;j++)
```

```
        {
```

```
            temp=0;
```

```
            for (k=i;k<=j;k++)
```

```
            {
```

```
                temp=temp+a[k];
```

```
            }
```

```
            if (max<temp) max=temp;
```

```
        }
```

```
    }
```

```
    cout<<max;
```

```
    return 0;
```

```
}
```

```
5
```

```
1 -2 3 1 -4
```

```
4
```

一座山上有 10 个山洞。一天，格莱尔和尼克在山上玩捉迷藏游戏。尼克说：“我先把 10 个山洞从 1~10 编上号，你从 10 号洞出发，先到 1 号洞找我第二次隔 1 个洞找我，第三次隔 2 个洞找我，如图 62.1 所示。以后以此类推，次数不限。”格莱尔同意了，但她从早到晚进洞 1000 次，也没找到尼克。

试编一程序，算一算兔子尼克可能躲在几号洞里

用数组 a 记录格莱尔进 10 个洞的情况，首先把 a[1] 至 a[10] 的值初始化为 true，表示格莱尔未进 1~10 号洞。当格莱尔进过 i 号洞时，a[i] 标记为 false。变量 cishu 表示进洞的次数，用“ $i=(i+cishu)\%10$ ”来确定每次格莱尔进的是哪个洞。最后，输出没进过的洞即可。流程图如图 62.2 和图 62.3

4.5-4

`#include<iostream>`// 选自潘洪波《小学生 c++ 趣味编程》第 62 课捉迷藏

`using namespace std;`

`int main()`

`{`

`bool a[11];`

`int i, cishu;`

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

`a[i]=true;`

`i=10;`

`a[i]=false;`

`cishu=1;`

`while (cishu<=1000)`

`{`

`i=(i+cishu)%10;`

`if (i==0) i=10;`

`a[i]=false;`

`cishu++;`

`}`

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

`if (a[i])`

`cout<<i<<endl;`

`return 0;`

`}`

2
4
7
9

4.5-5

#include<iostream>// 选自潘洪波《小学生 c++ 趣味编程》第 63 课老鹰捉小鸡

using namespace std;

int main()

```
{  
    int i, j, a[6], n;  
    for (i=1; i<6; i++)  
        a[i]=i;  
    // 输出第 1 次的位置  
    i=1;  
    cout<<i<<": ";  
    for (j=1; j<6; j++)  
        cout<<a[j]<<" ";  
    cout<<endl;  
  
    for (i=2; i<=10; i++)  
    {  
        // 移动位置  
        for (j=0; j<=4; j++)  
            a[j]=a[j+1];  
        a[5]=a[0];  
        // 输出位置  
        cout<<i<<": ";  
  
        for (j=1; j<=5; j++)  
            cout<<a[j]<<" ";  
        cout<<endl;  
    }  
    return 0;  
}
```

狐狸老师和格莱尔等 5 位小朋友玩老鹰捉小鸡游戏，狐狸老师当“老鹰”、排在第 1 位的小朋友当“母鸡”，其他 4 位小朋友当“小鸡”。但是“母鸡”很辛苦，所以过一段时间“母鸡”需要排到队伍最后成为“小鸡”，让第 2 位小朋友当“鸡”……

试编一程序，模拟 10 次住置的变化过程。

第 1 次的位置 1 2 3 4 5
第 2 次的位置 2 3 4 5 1
第 3 次的位置 3 4 5 1 2
第 4 次的位置 4 5 1 2 3
第 5 次的位置 5 1 2 3 4

如图 63.1 所示，用数组 a 保存位置，a[1] 至 a[5] 保存第 1 个位置至第 5 个位置上小朋友的编号，a[0] 在位置移动中起到“中转站”的作用，暂时保存要由 a[1] 移动到 a[5] 的编号。流程图如图 63.2 和 63.3 所示。

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

4.5-6

#include<iostream>// 选自潘洪波《小学生 c++ 趣味编程》第 69 课胡萝卜与骨头

```
using namespace std;

int main()
{
    int i, top, a[31], num=0, k=0;
    for (i=1; i<=30; i++)
        a[i]=0;
    top=1;
    i=top;
    while (num<15)
    {
        if (i>30) i=1;
        if (a[i]==0) k++;
        if (k==9)
        {
            a[i]=1;
            k=0;
            num++;
        }
        i++;
    }
    cout<<" 骨头所在的位置: ";
    for (i=1; i<=30; i++)
        if (a[i]==0)
            cout<<i<<" ";
    return 0;
}
```

尼克喜欢胡萝卜，格莱尔喜欢骨头。15 根胡萝卜和 15 根骨头排成一圈狐狸老师要求尼克从第一根开始按 1~9 数数，逢九取出，直到剩下 15 根骨头为止。

试编一程序，算一算这 15 根胡萝卜和 15 根骨头应该如何排列，才能使剩下的 15 根全是骨头。

用数组 a 的下标表示胡萝卜或骨头原来在圈中的位置，其元素 a[i] 值为 0 时表示仍在圈内，为 1 则表示已取出，用变量 num 表示已取出的数的个数变量 top 表示第 1 次开始数的位置，变量 k 的值 1~9 为所数的数。

每当数到 9 时 (k==9)，赋值 a[i]=1，表示已取出，同时将 k 赋值为 0，重新开始 1~9 数数。当取出的个数达到 15 个时 (num==15) 检索数组各元素的值，为 0 时表示仍在圈内，就输出它的编号流程图如图 69.1 和图 69.2 所示。

```
骨头所在的位置 :
1  2  3  4  10  11
13 14  15  17  20
21  25  28  29
```

二维数组

```
int h[4][5]
```

h[0][0]	h[0][1]	h[0][2]	h[0][3]	h[0][4]
h[1][0]	h[1][1]	h[1][2]	h[1][3]	h[1][4]
h[2][0]	h[2][1]	h[2][2]	h[2][3]	h[2][4]
h[3][0]	h[3][1]	h[3][2]	h[3][3]	h[3][4]

```
int a[2][3]={ {1, 2, 3}, {4, 5, 6} }
```

```
int a[2][3]={ 1, 2, 3, 4, 5, 6 }
```

以上两句相当于：

```
a[0][0]=1;a[0][1]=2;a[0][2]=3;
```

```
a[1][0]=4;a[1][1]=5;a[1][2]=6;
```

4.6-1

```
#include<iostream>
```

```
using namespace std;
```

```
int main()
```

```
{//3 名学生 4 科成绩：小明，小红，小文；语文数学，英语，科学
```

```
float c[3][4]=
```

```
{
```

```
    {81, 82, 83, 84},
```

```
    {71, 72, 73, 74},
```

```
    {91, 92, 93, 94},
```

```
};
```

```
cout<<c[0][0]<<endl;// 小明语文成绩
```

```
cout<<c[1][2];// 小红英语成绩
```

```
return 0;
```

```
}
```

81

73

默认下标从 0 开始

4.6-2

```
#include<iostream>
using namespace std;
int main()
{
    float c[5][5];
    int i, j;
    for (i=0; i<3; i++)
    {
        for (j=0; j<4; j++)
        {
            cin>>c[i][j];
        }
    }
    cout<<c[0][0]<<endl; // 小明语文成绩
    cout<<c[1][2]; // 小红英语成绩
    return 0;
}
```

```
81 82 83 84
71 72 73 74
91 92 93 94
81
73
```

4.6-3

```
#include<iostream>
using namespace std;
int main()
{
    float c[5][5];
    int i, j;
    for (i=1; i<=3; i++)
    {
        for (j=1; j<=4; j++)
            cin>>c[i][j];
    }
    cout<<c[1][1]<<endl; // 小明语文成绩
    cout<<c[2][3]; // 小红英语成绩
    return 0;
}
```


矩阵练习题

4.7-1

```
#include<iostream>// 杨辉三角
#include<cstring>
#define maxn 110//define 宏定义
using namespace std;
int main()
{
    int i, j, n, c[maxn][maxn];
    cin>>n;
    memset(c, 0, sizeof(c));// 数组初始化为 0
    c[0][0]=1;
    c[1][0]=1;
    c[1][1]=1;// 赋初值
    for (i=2; i<n; i++)
    {
        c[i][0]=1;
        for (j=1; j<=i; j++)
        {
            c[i][j]=c[i-1][j-1]+c[i-1][j];
        }// 每个数字是它左上方和上方的数字求和
    }
    for (i=0; i<n; i++)
    {
        for (j=0; j<=i; j++)
        {
            cout<<c[i][j]<<" ";
        }
        cout<<endl;
    }
    return 0;
}
```

```
8
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

4.7-2

```
#include<iostream>// 数字三角形
```

```
#include<iomanip>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
    int a[101][101];
```

```
    int n, i, j;
```

```
    cin>>n;
```

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

```
    {
```

```
        for (j=i; j<=n; j++)
```

```
        {
```

```
            a[i][j]=j-i+1;
```

```
        }
```

```
    }
```

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

```
    {
```

```
        for (j=1; j<=n; j++)
```

```
        {
```

```
            if (!a[i][j])
```

```
                cout<<setw(5)<<" "; // 如果不存在数据, 输出空格
```

```
            else
```

```
                cout<<setw(5)<<a[i][j];
```

```
        }
```

```
        cout<<endl;
```

```
        cout<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

```
5
  1  2  3  4  5
    1  2  3  4
      1  2  3
        1  2
          1
```

4.7-3

```
#include<iostream>// 回形方阵
```

```
using namespace std;
```

```
int main()
```

```
{  
    int n, i, j, k, a[100][100];  
    cin>>n;  
    for (k=1;k<=(n+1)/2;k++)  
    {  
        for (i=k;i<=(n+1)-k;i++)  
        {  
            for (j=k;j<=(n+1)-k;j++)  
            {  
                a[i][j]=k;  
            }  
        }  
    }  
  
    for (i=1;i<=n;i++)  
    {  
        for (j=1;j<=n;j++)  
        {  
            cout<<a[i][j]<<" ";  
        }  
        cout<<endl;  
    }  
    return 0;  
}
```

```
8  
1 1 1 1 1 1 1 1  
1 2 2 2 2 2 2 1  
1 2 3 3 3 3 2 1  
1 2 3 4 4 3 2 1  
1 2 3 4 4 3 2 1  
1 2 3 3 3 3 2 1  
1 2 2 2 2 2 2 1  
1 1 1 1 1 1 1 1
```

4.7-4-a

```

#include<cstdio> // 螺旋方阵
#include<iostream> // 本题选自《ccf 中学生计算机程序设计》
using namespace std;
int main()
{
    int i, j, n, t, count=1;
    int num[26][26];
    cin>>n;
    for (i=1; i<=n; i++) //memset (num, 0, sizeof (num)); // 数组初始化为 0, 另一种写法
    {
        for (j=1; j<=n; j++)
            num[i][j]=0;
    }
    i=1; j=n; // 第一个数字的下标
    num[i][j]=count;
    for (t=1; t<=n*n; t++)
    {
        //!num[i+1][j] 意思是 num[i+1][j] 数值为 0
        while (i+1<=n&&!num[i+1][j]) // 向下写
            num[++i][j]=++count;

        while (j-1>=1&&!num[i][j-1]) // 向左写
            num[i][--j]=++count;

        while (i-1>=1&&!num[i-1][j]) // 向上写
            num[--i][j]=++count;

        while (j+1<=n&&!num[i][j+1]) // 向右写
            num[i][++j]=++count;
    }
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j++)
            printf ("%4d", num[i][j]); // 场宽为 4
        cout<<endl;
    }
    return 0;
}

```

4	10	11	12	1
	9	16	13	2
	8	15	14	3
	7	6	5	4

4.7-4-b

```
#include <iostream> // 螺旋方阵
#include <iomanip>
using namespace std;
int s[1000][1000];
int main() {
    int n, i, j, sum=1;
    cin>>n;
    i=1, j=0; // 第一个数据下标
    while (sum<=n*n)
    {
        while (s[i][++j]==0&&j<=n) // 向右, 如果下个数为0, 且没有超出矩阵边
            s[i][j]=sum++;
        j--; // 如果向右到边, 向下

        while (s[++i][j]==0&&i<=n) // 向下
            s[i][j]=sum++;
        i--; // 如果向下到边, 向左

        while (s[i][--j]==0&&j>=1) // 向左
            s[i][j]=sum++;
        j++; // 如果向左到边, 向上

        while (s[--i][j]==0&&i>=1) // 向上
            s[i][j]=sum++;
        i++; // 如果向上到边, 向右
    }
    for (i=1; i<=n; i++)
    {
        for (j=1; j<=n; j++)
            cout<<setw(3)<<s[i][j]; //setw() 函数作用是字符宽度设置
        cout<<endl;
    }
    return 0;
}
```

```
4
 1  2  3  4
12 13 14  5
11 16 15  6
10  9  8  7
```

4.7-5

```
#include<iostream>// 蛇形方阵
```

```
#include<iomanip>
```

```
using namespace std;
```

```
int main() {
```

```
    int n, i, j, k, t=0, a[21][21];
```

```
    cin>>n;
```

```
    for (k=1; k<=n; k++)
```

```
    {
```

```
        if (k%2)
```

```
        {
```

```
            for (j=1; j<=k; j++)
```

```
            {    i=k+1-j;
```

```
                t++;
```

```
                a[i][j]=t;
```

```
                a[n+1-i][n+1-j]=n*n+1-t;
```

```
            }
```

```
        }
```

```
        else
```

```
        {
```

```
            for (j=k; j>=1; j--)
```

```
            {    i=k+1-j;
```

```
                t++;
```

```
                a[i][j]=t;
```

```
                a[n+1-i][n+1-j]=n*n+1-t;
```

```
            }
```

```
        }
```

```
    }
```

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

```
    {
```

```
        for (j=1; j<=n; j++)    cout<<setw(5)<<a[i][j];
```

```
        cout<<endl<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

5

1 2 6 7 15

3 5 8 14 16

4 9 13 17 22

10 12 18 21 23

11 19 20 24 25

4.7-6// 本题及其以下选自《课课通》

```
#include<iostream>//n 阶奇数幻方
```

```
#include<iomanip>
```

```
#include<cstring>
```

```
using namespace std;
```

```
int main() {
```

```
    int n, i, j, k, a[21][21];
```

```
    memset(a, 0, sizeof(a)); // 初始化数组, 都为 0
```

```
    cin>>n;
```

```
    i=1; j=n/2+1;
```

```
    a[i][j]=1;
```

```
    for (k=2; k<=n*n; k++)
```

```
    {
```

```
        if (k%n==1) i++; // 如果排满一行, 则开始排下一行
```

```
        else
```

```
        {
```

```
            i--;
```

```
            j++;
```

```
            if (i==0) i=n; // 如果超出行数, 从最后一行开始排列数字
```

```
            if (j==n+1) j=1; // 如果超出列数, 从第一个开始排列数字
```

```
        }
```

```
        a[i][j]=k;
```

```
    }
```

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

```
    {
```

```
        for (j=1; j<=n; j++)
```

```
            cout<<setw(5)<<a[i][j];
```

```
        cout<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

5

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

搜索基础

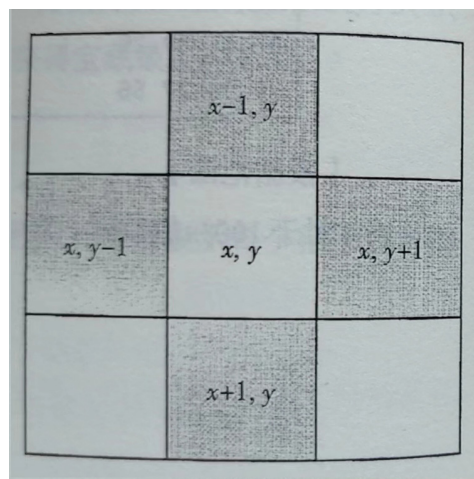
本章节主要选自《信息学竞赛金牌导航》

#include<bits/stdc++.h>//4.8-1 一维数组四向走法

```
using namespace std;
int dx[4]={-1,0,1,0};
int dy[4]={0,1,0,-1};
int main() {
    int x,y,i,nx,ny;
    cin>>x>>y;
    for(i=0;i<4;i++)
    {
        nx=dx[i]+x;
        ny=dy[i]+y;
        cout<<nx<<" "<<ny<<endl;
    }
    return 0;
}
```

#include<bits/stdc++.h>// 二维数组四向走法

```
using namespace std; //4.8-2
int dxy[4][2]={{-1,0},{0,1},{1,0},{0,-1}};
int main() {
    int x,y,i,nx,ny;
    cin>>x>>y;
    for(i=0;i<4;i++)
    {
        nx=dxy[i][0]+x;
        ny=dxy[i][1]+y;
        cout<<nx<<" "<<ny<<endl;
    }
    return 0;
}
```



如果把二维数组比作棋盘或者地图，则方向控制数组就是对应的向量，用于描述一个位置到另一个位置的方向与距离。

在棋盘中，士兵的移动规则常被设定为“单次只能移动到同行或同列的相邻格”。转述为下标关系则是从位置 (x, y) 可以向上、下、左、右四个方向推进一格。

若某个士兵从位置 (x, y) 出发，只移动一步，则可以落脚灰色格子：向上走可到 $(x-1, y)$ ，向右走可到 $(x, y+1)$ ，向下可走到 $(x+1, y)$ ，向左可走到 $(x, y-1)$ 。

输入：

2 5

输出：

1 5

2 6

3 5

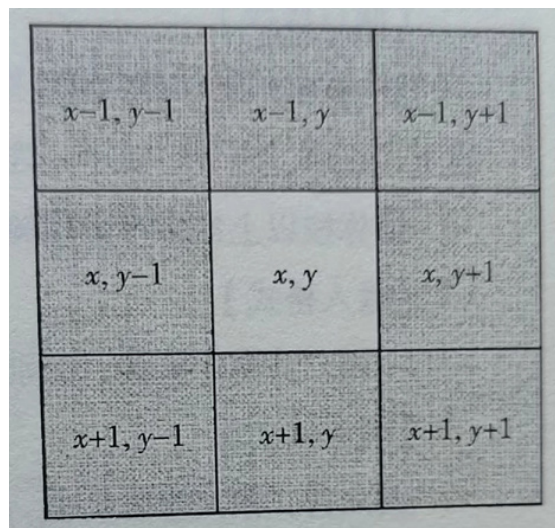
2 4

多向控制数组

在国际象棋中，王的移动规则被设定为“单次只能移动落到相邻的格子中（八选一）”。转述为下标关系则是：从位置 (x, y) 可以选择向上、下、左、右、左上、右上、左下、右下八个方向中的任意一个方向推进一格。

若王从位置 (x, y) 出发，只移动一步，则可以落脚到灰色格子区域：向上走可到 $(x-1, y)$ ，向右走可到 $(x, y+1)$ ，向下可走到 $(x+1, y)$ ，向左可走到 $(x, y-1)$ ，向左上可走到 $(x-1, y-1)$ ，向右上可走到 $(x-1, y+1)$ ，向右下可走到 $(x+1, y+1)$ ，向左下可走到 $(x+1, y-1)$ 。

从 (x, y) 位置偏移 to 相邻格子，其下标的增量由八向的方向控制数组描述。



4.8-3

`#include<bits/stdc++.h> // 二维数组八向走法`

`using namespace std;`

`int dxy[8][2]={{-1, 0}, {-1, 1}, {0, 1}, {1, 1}, {1, 0}, {1, -1}, {0, -1}, {-1, -1}};`

`int main()`

`{`

`int x, y, i, nx, ny;`

`cin>>x>>y;`

`for (i=0; i<8; i++)`

`{`

`nx=dxy[i][0]+x;`

`ny=dxy[i][1]+y;`

`cout<<nx<<" "<<ny<<endl;`

`}`

`return 0;`

`}`

输入:

5 5

输出:

4 5

4 6

5 6

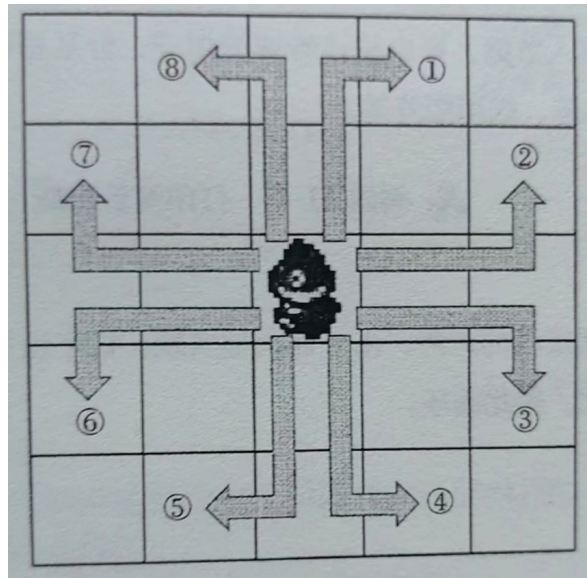
6 6

6 5

6 4

5 4

4 4



4.8-4

`#include<bits/stdc++.h> // 二维数组马的走法`

`using namespace std;`

`int dxy[8][2]={{-2,1},{-1,2},{1,2},{2,1},{2,-1},{1,-2},{-1,-2},{-2,-1}};`

`int main()`

`{`

`int x,y,i,nx,ny;`

`cin>>x>>y;`

`for(i=0;i<8;i++)`

`{`

`nx=dxy[i][0]+x;`

`ny=dxy[i][1]+y;`

`cout<<nx<<" "<<ny<<endl;`

`}`

`return 0;`

`}`

输入:

5 5

输出:

3 6

4 7

6 7

7 6

7 4

6 3

4 3

3 4

4.8-5

```
#include<bits/stdc++.h> // 环形矩阵
using namespace std;
int a[300][300];
int n,m;
int dx[4]={0,1,0,-1};
int dy[4]={1,0,-1,0};
int main()
{
    int i,j;
    cin>>n;
    memset(a,-1,sizeof(a)); //a 数组初始化为 -1
    for (i=1;i<=n;i++) // 图形部分设置为 0
    {
        for (j=1;j<=n;j++) a[i][j]=0;
    }
    int x=1,y=1,st=1,t=0;
    while (st<=4*n-4)
    {
        a[x][y]=st++; // 本句等于 a[x][y]=st; st++;
        x=x+dx[t];
        y=y+dy[t];
        if (a[x+dx[t]][y+dy[t]]!=0)
        {
            t=(t+1)%4; // 求余数是为了图形拐弯
        }
    }
    for (i=1;i<=n;i++) // 输出
    {
        for (j=1;j<=n;j++)
            cout<<setw(5)<<a[i][j];
        cout<<endl<<endl<<endl;
    }
    return 0;
}
```

5	1	2	3	4	5
16	0	0	0	0	6
15	0	0	0	0	7
14	0	0	0	0	8
13	12	11	10	9	

4.8-6

```
#include<bits/stdc++.h> // 螺旋矩阵
```

```
using namespace std;
```

```
int a[300][300];
```

```
int n, m;
```

```
int dx[4]={0, 1, 0, -1};
```

```
int dy[4]={1, 0, -1, 0};
```

```
int main()
```

```
{
```

```
    int i, j;
```

```
    cin>>n>>m;
```

```
    memset(a, -1, sizeof(a)); //a 数组初始化为 -1
```

```
    for (i=1; i<=n; i++) // 图形部分设置为 0
```

```
    {
```

```
        for (j=1; j<=m; j++) a[i][j]=0;
```

```
    }
```

```
    int x=1, y=1, st=1, t=0;
```

```
    while (st<=n*m)
```

```
    {
```

```
        a[x][y]=st++;
```

```
        x=x+dx[t];
```

```
        y=y+dy[t];
```

```
        if (a[x+dx[t]][y+dy[t]]!=0)
```

```
        {
```

```
            t=(t+1)%4; // 求余数是为了图形拐弯
```

```
        }
```

```
    }
```

```
    for (i=1; i<=n; i++) // 输出
```

```
    {
```

```
        for (j=1; j<=m; j++)
```

```
            cout<<setw(5)<<a[i][j];
```

```
        cout<<endl<<endl<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

5	5				
	1	2	3	4	5
	16	17	18	19	6
	15	24	25	20	7
	14	23	22	21	8
	13	12	11	10	9

4.8-7

```
#include<bits/stdc++.h> // 蛇形矩阵
```

```
using namespace std;
```

```
int a[300][300];
```

```
int n,m;
```

```
int dx[4]={1,-1,0,1};
```

```
int dy[4]={0,1,1,-1};
```

```
int main() {
```

```
    int i,j;
```

```
    cin>>n>>m;
```

```
    memset(a,-1,sizeof(a)); //a 数组初始化为 -1
```

```
    for (i=1;i<=n;i++) // 图形部分设置为 0
```

```
    {
```

```
        for (j=1;j<=m;j++) a[i][j]=0;
```

```
    }
```

```
    int x=1,y=1,st=1,t=0;
```

```
    a[x][y]=st; // 开始设置的数字
```

```
    while (st<=n*m)
```

```
    {
```

```
        x=x+dx[t];
```

```
        y=y+dy[t];
```

```
        if (a[x][y]==0) a[x][y]=++st;
```

```
        if (a[x+dx[t]][y+dy[t]]!=0 || t==0 || t==2)
```

```
        {
```

```
            t=(t+1)%4;
```

```
        }
```

```
    }
```

```
    for (i=1;i<=n;i++) // 输出
```

```
    {
```

```
        for (j=1;j<=m;j++)
```

```
            cout<<setw(5)<<a[i][j];
```

```
        cout<<endl<<endl<<endl;
```

```
    }
```

```
    return 0;
```

```
}
```

3	4		
1	3	4	9
2	5	8	10
6	7	11	12