晖 爸 编 程
PROGRAMMING

信奥编程讲义

（3 学阶）

陈琰宏　编著

微信：13989476922

邮箱：zufe_graphics@163.com

目　　录

第一讲 一维数组

重点

1. 理解数组的含义。
2. 学会一维数组的定义。
3. 掌握一维数组的元素引用和物理存储方式
4. 理解为什么要用数组，数组的定义、初始化、引用。

思考：为什么要使用数组？

例：输入 50 个学生的某门课程的成绩，打印出低于平均分的学生序号与成绩。

[分析] 因为只有读入最后一个学生的分数后才能求得平均分，并且要求打印出低于平均分的学生序号和成绩，故必须把 50 个学生的成绩都保留起来，然后逐个和平均分比较，把低于平均分的成绩打印出来。如果，用简单变量 a1，a2,…，a50 存储这些数据，要用 50 个变量保存输入的数据，程序片断如下：

```
cin>>a1>>a2>>…>>a10…cin>>a41>>a42>>…>>a50;
```

可以看出，这样的程序是多么繁琐。**如果说处理的数据规模达到成千上万，上面的例子单单读入就会异常复杂，电脑的优势没有得到体现**。从以上的讨论可以看出，如果只使用简单变量处理大量数据，就必须使用大量只能单独处理的变量，即使是简单问题也需要编写冗长的程序。

我们需要**把一大批具有相同性质的数据组合成一个新类型的变量**，可以用简单的程序（比如循环 50 次）对这个新变量的各个分量进行相同的处理，每个分量仍然保留单个变量的所有性质。像数学中使用下标变量 ai 形式表示这 50 个数，则问题就容易实现。在 C++语言中，具有下标性质的数据类型是数组。

例如，读入 50 个学生的成绩，只需写如下语句即可：

```
for (int i=1;i<=50;++i)
     cin>>a[i];
```

例 1：输入 50 个学生的某门课程的成绩，打印出低于平均分的学生序号与成绩。

```
tot = 0;
for (int i=1;i<=50;++i)
{
  cin>>a[i];
   tot+=a[i];
}
ave= tot/50;                          //计算平均分
for (int i=1;i<=50;++i)
  if (a[i]<ave) cout<<"No. "<<i<<" "<<a[i];
```

1.1 数组的定义

定义一维数组的格式如下：

类型标识符 数组名 [常量表达式];

在定义数组时必须指定数组的类型，与普通变量的定义类似，一个数组在内存中占一片连续的存储单元。其中，类型标识符可以是任何基本数据类型，相同类型的数组可以一起定义。数组名必须是合法的标识符。常量表达式的值即为数组元素的个数。

例 int a[10];

用数组名代表这一批数据，用下标来区分这些数据。这样处理既可以省略很多个变量名，使得程序精炼；又便于对这些数据作统一处理。

说明：

- 数组名也是一个标识符，必须遵循标识符命名规则。
- 数组名表示整个数组所占存储空间的首地址(其实也是第 0 个元素的首地址)，是一个表示地址的常量，它的值是不能改变的，比如“a++”就是非法的。
- 用方括号括起来的常量表达式表示数组元素个数，即数组长度。如下面的写法是合法的：
- 数组元素的下标从 0 开始，例如定义了“int a[10]；”，则这 10 个元素是： a[0]、a[1]、a[2]、a[3]、a[4]、a[5]、a[6]、a[7]、a[8]、a[9]。注意最后一个元素是 a[9]，而不是 a[10]。
- 常量表达式中可以包括常量、常变量和符号常量，但不能包含变量。也就是说，C++不允许对数组的大小作动态定义。

1.2 数组的使用

1. 引用一维数组的元素

数组必须先定义，再使用。只能逐个引用数组元素而不能一次性引用整个数组中的全部元素。

```
int a[10];
a[0] = a[5] + a[7] - a[2*3];
int i;
for( i=0; i<10; i++ )
    scanf( "%d", &a[i] );   //cin>>a[i]; 输入 10 个数据到数组中
for( i=0; i<10; i++ )  printf( "%d", a[i] );    //输出 10 个数组元素的值
```

printf("%d", a); //试图一次把 10 个数组元素的值输出来，这样写是错误的

填空：输入具有 n 个元素的数组 _____

2. 一维数组的初始化

所谓初始化，就是在定义数组时，就给数组元素赋值。对数组元素的初始化可以采用以下方法实现：

1)在定义数组时分别对数组元素赋予初值。例如：

```
int a[10]={0，1，2，3，4，5，6，7，8，9};
```

2) 可以只给一部分元素赋值。例如：

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

3) 在对全部数组元素赋初值时，可以不指定数组长度。例如：

```
int a[5]={1，2，3，4，5};
```

可以写成：

int a[ ]={1，2，3，4，5};

4) 对数组元素全部初始化为0，可以简写为：{0}。

例如：int a[5]={0}; 将数组a的5个元素都初始化为0。

说明：数组不初始化，则其元素值为随机数。

5) 给数组“整体”赋值

- **memset 函数** memset 函数是给数组“按字节”进行赋值，一般用在 char 型数组中，如果是 int 类型的数组，一般赋值为 0 和 -1。使用前需要包含头文件：#include <cstring>。
- **fill 函数** fill 函数是给数组“按元素”进行赋值，可以是整个数组，也可以是部分连续元素，可以赋任何值。使用前需要包含头文件：#include <algorithm>。

填空：把数组a[20]的每个元素都赋值为0 _____

把数组b[20]的每个元素都赋值为5 _____

例1 阅读并上机调试程序，体会 memset 和 fill 函数的作用。

```
#include<iostream>
#include<cstring>
using namespace std;
int main(){
    int a[10],b[10],c[10],d[10],i;
    memset(a,0,sizeof(a));   // 将a数组所有元素均赋值为0
    for(i = 0; i < 9; i++) cout << a[i] << " "  ;
    cout << a[9] << endl;
    memset(c,0,5);
    //将 c 数组前 5 个字节都赋值为 0，所以只能确定 c[0]等于0，其他元素值不确定
    for(i = 0; i < 9; i++) cout << c[i] <<" " ;
    cout << c[9] << endl;
    fill(d,d+5,8);//将 d 数组前 5 个元素都赋值为 8，其他元素值不确定
    for(i = 0; i < 9; i++) cout << d[i] <<" " ;
    cout << d[9] << endl;
    return 0;
}
```

1.3 一维数组应用

1 [7784] 奇数输出

输入n个数，然后按照原来的顺序输出这n个数中的奇数

输入 第一行 n（n<1000） 第二行 n个数

输出 按照原来的顺序输出这n个数中的奇数

样例输入

10
9 8 5 6 7 8 7 2 3 4
样例输出
9 5 7 7 3

思路：

1. 定义一个具有 1000+x 个元素的数组_____
2. 将 n 个元素输入_____
3. 进行奇数判断求和_____

```
#include<iostream>
using namespace std;
int main()
{
    int i,n,a[1001];
    cin>>n;
    for(i=0;i<n;i++)
    {
      cin>>a[i];
       if(a[i]%2==1)
       cout<<a[i]<<" ";
    }
    return 0;
}
```

思考：此题可以不用数组做吗？为什么？

2［7785］小明存钱

题目描述
小明要存钱过年买东西，现在已知他已经存了 m 个月，还有每个月存的金额。小明想知道一共存了多少钱。
输入
第一行，一个数 m 表示存钱的月数(m<=12)。 第二行，m 个数分别表示每个月的存钱数，数中间空格隔开。
输出
一个数表示存钱总和。
样例输入
4
100 150 50 200
样例输出
500

```
include<iostream>
using namespace std;
int main()
```

```
{
    int i,n;
    __1_____________________
    cin>>n;
    for(i=0;i<n;i++)
    {
        __2_____________________
        __3_____________________
    }
    cout<<sum;
    return 0;
}
```

3［3317］求平均值 2

输入 n（n<=20）个学生的某门课程的成绩，打印出平均分和大于平均分的学生人数。小数点后面保留两位。

样例输入

5

90 88 76 91 67

样例输出：82.40 3

思路：循环输入各个学生的成绩，求总分、再求平均值；然后求人数。

```
#include<iostream>
using namespace std;
int main(){
        int n,a[10],sum=0,k=0;
        double ave;
        cin>>n;
        for(int i=1;i<=n;i++){
                cin>>a[i];
                sum=sum+a[i];
        }
        ave=1.0*sum/n;
        for(int i=1;i<=n;i++)
        if(a[i]>ave)
        k++;
        cout.precision(2);
        cout<<fixed<<ave<<" "<<k<<endl;
        return 0;
}
```

思考：此题可以不用数组做吗？为什么？

4 [2663] 陶陶摘苹果

陶陶家的院子里有一棵苹果树，每到秋天树上就会结出 10 个苹果。苹果成熟的时候，陶陶就会跑去摘苹果。陶陶有个 30 厘米高的板凳，当她不能直接用手摘到苹果的时候，就会踩到板凳上再试试。 现在已知 10 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够摘到的苹果的数目。假设她碰到苹果，苹果就会掉下来。

思路：_____

```
#include<iostream>
using namespace std;
int main (){
    int __1_____________;
    int h,sum=0;
    for(int i=0;i___2_______;i++){
        __3____________
    }
    cin>>h;
    for(int k=0;k<10;k++){
        if(_4_____________)sum++;

    }
    cout<<sum;
    return 0;
}
```

5 [2664] 计算书费

给定 10 种图书的单价以及每种图书购买的数量，编程计算应付的总费用。

思考：本题的核心是如何实现数组元素的初始化？

```
 double b[11]={_____________________________________};

#include<iostream>
#include<cmath>
using namespace std;
int main (){
    int a[_________];
    double b[11]={28.9,32.7,45.6,78,35,86.2,27.8,43,56,65};
    double sum=0;
    for(int i=0;i<10;i++){

        _________________
    }
    for(int k=0;k<10;k++){
        sum=sum+_________________;
```

```
    }
    cout.precision(1);
    cout<<fixed<<sum;
    return 0;
}
```

6 *[2256] 斐波那契数列

求斐波那契数列项。斐波那契数列的定义为：
f(1)=1, f(2)=1
f(n)=f(n-2)+f(n-1)　　(n>1)
序列为 1 1 2 3 5 8 13　.....输入某一项（n>2）输出该项对应的序列值
样例输入 6
样例输出 8

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[50]={_________},i,n;
    scanf("%d",&n);
    for(i=_______;i<=n;i++){
        a[i]=__________;
    }
    printf("%d",________);
    return 0;
}
```

7 *[2665] 数组逆序重存放

将一个数组中的值按逆序重新存放。例如，原来的顺序为 8,6,5,4,1。要求改为 1,4,5,6,8。输入为两行：第一行数组中元素的个数 n（1<n<100)，第二行是 n 个整数，每两个整数之间用空格分隔。输出为一行：输出逆序后数组的整数，每两个整数之间用空格分隔。
样例输入:
5
8 6 5 4 1
样例输出
1 4 5 6 8

```
#include <iostream>
using namespace std;
int main()
{
        int a[105],n,i;
        cin>>n;
        for(i=1;i<=n;i++)cin>>a[i];
```

```
    return 0;
}
```

作业列表

2222 输出平均值　　3317 求平均值 2
2663 陶陶摘苹果　　2664 计算书费
2665 数组逆序重存放　　2672 最大值和最小值的差
6884 求最大最小值　　1312 十佳歌手大赛
2256 斐波那契数列

习题

(1) int a[4]={5,3,8,9};其中 a[3]的值为(　D　)。
　　A. 5　　B. 3　　C. 8　　D. 9
(2) 设有如下程序段;
```
1 int  w[5]={ 1,2,4 }, i;
2 scanf("%d",&w);//cin>>w;
3 for( i=3;i<5;i++ ) w[i]=2*i+1;
4 printf("w[%d]=%d\n",4,w[4]);//cout<<"w["<<4<<"]="<<w[4]<<endl;
```
则以下选项中存在错误的是(　都对　)
　　A)第 1 行　　B)第 2 行　　C)第 3 行　　D)第 4 行
(3) 若有以下数组说明，则数值最小的和最大的元素下标分别是(　　)。
```
int a[12] ={1,2,3,4,5,6,7,8,9,10,11,12};
```
　　A. 1,12　　B. 0,11　　C. 1,11　　D. 0,12
(4) 若有初始化"int a[5]={1,2,3,4,5};",则值为 4 的表达式是(　　)
　　A)a[4]　　B)a[a[2]+1]　　C)a[a[2]]　　D)a[3]+1
(5) 若有定义"double a[3]={3.14,2.5,9.7},b=6;",则错误的赋值语句是(　　)
　　A)b=a[2];　　B)b=a+2.1];　　C)a[1]=b;　　D)b=a[0]+7;
(6) C++ 语言中，数组的各元素必须具有相同的 _____，元素的下标下限为 _____，下标必须是正整数、0、或者_____。但在程序执行过程中，不检查元素下标是否_____。
(7) 根据以下说明,写出正确的说明语句：men 是一个有 10 个整型元素的数组。_____
step 是一个有 4 个实型元素的数组,元素值分别为 1.9, -2.33, 0, 20.6。

_____

(8) 输入 n（n<=20）个学生的某门课程的成绩，打印出平均分和大于平均分的学生人数。

```
#include <iostream>
 using namespace std;
int main()
{       int  __1_____,n,i,k=0;
        double sum=0,ave;
        cin>>n;
        for(i=0;i<n;i++){
                2__________
                sum=sum+ 3      ;
        }
        ave=sum/n;
        for(i=0;i<n;i++)
                4_______________
        printf("%.2lf %d",ave,k);
          return 0;
}
```

第二讲 统计与查找

重点
1. 掌握一维数组的元素求最大值、最小值、求和等基础统计操作。
2. 掌握一维数组的元素查找。
3. 了解一维数组的二分查找。

一维数组的统计和查找包括在众多元素中与查找最大值、最小值、求和、求平均值，以及根据要求找到符合要求的元素等操作。

2.1 统计

1 [7792] 求最大值

输入 n(n<=1000)个数，输入 L 和 m，输出第 L 个和第 m 个之间的最大数。
输入包括两行，第一行输入 3 个数分别表示 n L m 的值，第二行 n 个数。
输出 n 个数中第 L 个和第 m 个之间的最大数
样例输入 复制
10 2 5
9 8 5 6 7 8 7 2 3 4
样例输出 复制
8

分析：先输入 n 个数，然后确定左右边界的值，再求此区间内的最大值。

```
#include <iostream>
using namespace std;
int main()
{
        int a[1000],m,i,j,max=0;
        cin>>m>>i>>j;
        for(int k=1;k<=m;k++)cin>>a[k];//1 输入
        for(int k=i;k<=j;k++){
                if(a[k]>max)max=a[k];//求解区间内最大值
        }
        cout<<max;
        return 0;
}
```

2 [6884] 求最大最小值

用数组存储从键盘上输入的 10 个整数，并求这 10 个整数的最大值、最小值和平均值。

```
#include<iostream>
```

```
using namespace std;
int main( )
{
    int a[___1____], i;
    int max, min;   //最大值、最小值
    double average; //平均值
    for( i=0; i<10; i++ )
        scanf( "%d", &____2_______ );
    max = min =____3______;
    average = a[0];
    for( ____4______; i++ )
    {
        if( a[i]>max )  _5__________;
        if( a[i]<min )  _6__________;
        average += a[i];
    }
    average = __7____________;
    printf( "max=%d, min=%d, average=%.1f\n", max, min, average );
}
```

3 [1312] 十佳歌手大赛

在十佳歌手大赛中，有一组评委给参赛选手打分（百分制）。选手最终得分的获取规则是：去掉一个评委给的最高分，去掉在一个评委给的最低分，取其余分数的平均值。需要说明的是，如果同时有多个评委给出最低分，那么只去掉一个最低分；同样的，如果同时有多个评委给出最高分，那么只去掉一个最高分。你的任务是设计一个程序，自动计算选手的得分。
输入
每个测试数据占一行：首先是一个正整数 N(2<N<20)，评委的个数；然后是 N 个百分制分数。
输出
对输入文件中的每个测试数据，输出该选手的得分，精确到小数点 2 位有效数字。

样例输入
5 98 90 95 88 85
样例输出
91.00

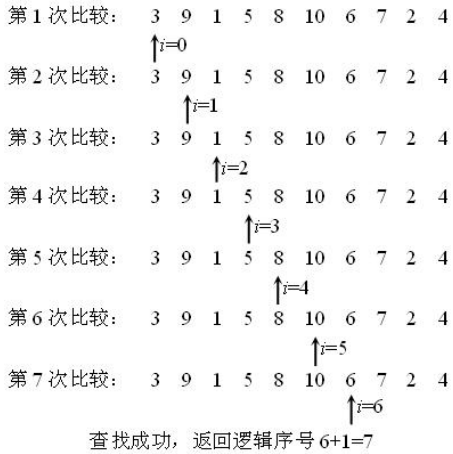
```
#include<iostream>
#include<cstring>
using namespace std;
int main(){
    int a[21],n,i;
    double ___1__________________;
    cin>>n;
    for(i=1;i<=n;i++){
        cin>>__2____________________;
```

```
        sum=sum+_3_________;
        if(a[i]>maxn)maxn=_4_______;
        if(a[i]<minn)_5___________;
    }
    ave=__6_____________;
    printf("%.2lf",ave);
    return 0;
}
```

4 [7795] 累加

输出一个整数数列中不与最大数相同的数字之和。
输入分为两行： 第一行为 N(N 为接下来数的个数，N <= 100)； 第二行为 N 个整数，数与数之间以一个空格分开，每个整数的范围是-1000,000 到 1000,000。
输出为 N 个数中除去最大数其余数字之和。
样例输入
3
1 2 3
样例输出
3
分析：先求出最大值，再访问每一个元素，

```
#include <iostream>
using namespace std;
int main()
{
        int a[1000],m,s=0,max=0;
        cin>>m;
        for(int k=1;k<=m;k++)cin>>a[k];
        for(int k=1;k<=m;k++){
                if(a[k]>max)max=a[k];
        }
        for(int i=1;i<=m;i++){
                if(a[i]!=max)s+=a[i];
        }cout<<s;
        return 0;
}
```



2.2 一维数组的查找

一维数组的查找操作，就是在一维数组中查找有没有某个元素，它的值等于指定的值 x。查找操作的结果可能是一个没找到、找到一个或者找到很多个。常见的查找算法有“顺序”查找和“二分”查找。

顺序查找是一种最简单的查找方法。它的基本思路是：从数组的一端开始顺序扫描，依次将扫描到的元素关键字和给定值 k 相比较，若当前扫描到的元素关键字与 k 相等，则查找成功；若扫描结束后，仍未找到关键字等于 k 的元素，则查找失败。

例 1 在关键字序列为{3,9,1,5,8,10,6,7,2,4}的顺序表采用顺序查找方法查找关键字为 6 的元素。顺序查找过程如下。

5 [6857]抽奖 1

公司举办年会，为了活跃气氛，设置了摇奖环节。参加聚会的每位员工都有一张带有号码的抽奖券。现在，主持人依次公布 n 个不同的获奖号码，小谢看着自己抽奖券上的号码 num，无比紧张。请编写一个程序，如果小谢获奖了，请输出他中的是第几个号码；如果没有中奖，请输出 0。

[输入格式]

第一行一个正整数 n，表示有 n 个获奖号码，2<n≤100。第二行包含 n 个正整数，之间用一个空格隔开，表示依次公布的 n 个获奖号码。第三行一个正整数 num，表示小谢抽奖券上的号码。1≤获奖号码，num<10000。

[输出格式]

一行一个整数，如果小谢中奖了，表示中奖的是第几个号码；如果没有中奖，则为 0。

[输入样例]

7

17 2 3 4 9555 6 1

3

[输出样例]

3

```
#include<cstdio>
using namespace std;
int main(){
    int n,i,num,f,g[101];
    scanf( " %d" ,&n);//cin>>n;
    for(i = 1; i <= n; i++) scanf( " %d " ,&g[i]);//cin>>g[i];
    scanf( " %d " ,&num);//cin>>num;
    f = 0;
    for(i = 1; i <= n; i++)
        if(g[i] == num){f = i; break;}
    printf( "%d\n " ,f);//cout<<f<<endl;
    return 0;
}
```

6 [2662] 相同的个数

输出一个整数序列中与指定数字相同的数的个数。

输入包含三行：第一行为 N，表示整数序列的长度(N ≤ 100)；第二行为 N 个整数，整数之间以一个空格分开；第三行包含一个整数，为指定的数字 m。

输出输出为 N 个数中与 m 相同的数的个数。

样例输入

3

2 3 2

2

样例输出

2

```
#include <iostream>
using namespace std;
int main()
{
        int a[105],n,i,x,k=0;
        cin>>n;
        for(i=1;i<=n;i++)cin>>a[i];
        cin>>x;
        for(i=1;i<=n;i++){
              ____________________
        }
        cout<<k;
          return 0;
}
```

7 *[6879] 比身高

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**。

[输入格式]

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。

[输出格式]

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。

```
#include<cstdio>
using namespace std;
int h[1001],n,i,j,ans,t1,t2;
```

```
int main(){
    scanf( "%d" ,&n);
    for(i = 1; i <= n; i++) scanf( "%d" ,&h[i]);
    for(i = 1; i <= n; i++){
        t1 = t2 = 0;
        for(j = 1; j < i; j++)
           if(h[j] > h[i]) t1++;// 排在他前面且比他高的人数
           for(j = i + 1; j <= n; j++)
              if(h[j] > h[i]) t2++;// 排在他后面且比他高的人数
           if(t1 == t2) ans++;
    }
    printf( " %d\n" ,ans);
    return 0;
}
```

作业列表

[7792] 求最大值	[6884] 求最大最小值
[1312] 十佳歌手大赛	[7795] 累加
[6857]抽奖 1	[2662] 相同的个数
[6879] 比身高	[7786] 简单查找
[7797] 统计 1	[7798] 统计 2

课前练习

1.

```
#include <iostream>
using namespace std;
int main()
{
    int sum=0,maxn,i;
    cin>>maxn;
    for(i=1; i<=maxn; i++)
    {
        if(i%2!=0)
            sum=sum+i;
    }
    cout<<sum<<endl;
    return 0;
}
```

输入：200

输出：10000

2

```
#include<iostream>
using namespace std;
int main(){
        int max=0,min=99999,m,n,i;
        cin>>m;
        for(i=0;i<m;i++)
        {
          cin>>n;
          if(n>max)max=n;
          if(n<min)min=n;
        }
        cout<<max-min;
        return 0;
}
```

输入：

5

	2 5 7 4 2 输出： 5

习题

1.有以下程序：

```
main( )
{  int  a[3]={ 1,2,3 },i;
   for ( i=3; i>=1; i-- ) printf("%d  ",a[i]);
}
```

程序运行后的输出结果是(B)

A)1 2 3 B)不确定的值 C)程序出错 D)3 2 1

2. 执行以下程序段后，x[2]的值是(A)

```
int  x[10]={ 1,2,3,4,5,6,7,8,9,10 },i,j,t;
i=0;  j=9;
while( i<j )
{  t=x[i];  x[i]=x[j]; x[j]=t;
   i++;  j--;
}
```

A)8 B)2 C)3 D)9

3. array 是一个一维整形数组,有 10 个元素,前 6 个元素的初值是 9,4,7,49,32,-5,正确的说明语句为：_____。该数组下标的取值范围是从___到___(从小到大)。用 scanf 函数输入数组的第二个元素表示为：_____。用赋值语句把 39 存入第一个元素表示为：_____。

把第六个和第四个元素之和存入第一个元素表示为：_____。

4. 若有以下整型的 a 数组，数组元素和它们得值如下所示:

数组元素: a[0] a[1] a[2] a[3] a[4] a[5] a[6] a[7] a[8] a[9]

元素的值: 9 4 12 8 2 10 7 5 1 3

①请写出对该数组的说明，并赋以上初值。_____

②该数组的最小下标值为___,最大下标值为___。

③写出下面各式的值：a[a[9]]的值为___ ；a[a[4]+a[8]]的值为___。

5.

```
#include<iostream>
using namespace std;
int main(){
        int a,max=-1000000,m,sum=0;
        cin>>m;
        for(int i=1;i<=m;i++)
        {
                cin>>a;
                if(a>=max) max=a;
                sum=sum+a;
        }
        sum=sum-max;
        cout<<sum<<endl;
        return 0;
}
```

输入 3 1 2 3

输出 3

6.

```
#include <bits/stdc++.h>
using namespace std;
int a[10001],n,i,k=1,maxn=0;
int main()
{
        cin>>n;
        a[0]=-34567;
        for(i=1;i<=n;i++)
        {
            cin>>a[i];
            if(a[i]==a[i-1])k++;
            else
                k=1;
            if(k>maxn) maxn=k;
        }
        cout<<maxn;
        return 0;
}
```

输入：

10

1 2 2 3 3 3 4 5 5 6

输出 3

第三讲 添加与删除

重点

1. 掌握一维数组的元素插入和删除操作。
2. 掌握一维数组的元素查找操作。

2.1 一维数组的插入

插入一个元素，需要先找到插入的位置(假设下标为 x)，将这个元素及其之后的所有元素依次往后移一位(注意要从后往前进行操作)，再将给定的元素插入(覆盖)到位置 x，如图 1 所示。

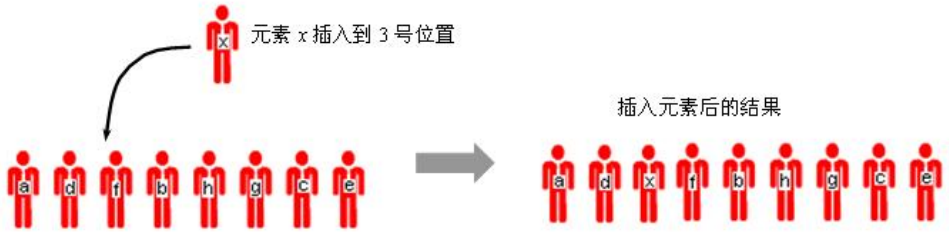


图 1 元素插入示意图

插入元素的步骤：

1. 移动。如图所示，要在 f 处插入元素 x，则要把 f~e 的所有元素右移，假设 f=3, n=8，则

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

2. 插入： a[f]=x;

1 [7805]插入 3

输入 10 个元素，然后输入 m，要求将 5 插入到原来数列的 m 位置中（解释：如果 m 为 3，则数字 5 就插入到第三个位置，原来的 3 号就变成了 4 号，原来的 4 号变 5 号……以此类推，数字位置从 1 开始计算）

输入两行 ,第一行 10 个元素 ，第二行 m（m 保证<=10）。

输出插入后的数列。

样例输入

5 8 9 5 6 4 3 2 6 10

3

样例输出

5 8 5 9 5 6 4 3 2 6 10

分析：将 [m,10]区间的元素后挪，然后插入大 m 位置即可。

```
#include<iostream>
#include<iostream>
using namespace std;
int main(){
   int a[12],m,i;
   for(i=1;i<=10;i++){
      cin>>a[i];
   }
   cin>>m;
   for(i=10;i>=m;i--){
      a[i+1]=a[i];
  }
   a[m]=5;
   for(i=1;i<=11;i++){
      cout<<a[i]<<" ";
   }
   return 0;
}
```

2［1447］插入元素

已有一个已排好的 9 个元素的数组，今输入一个数要求按原来排序的规律将它插入数组中。

输入第一行，原始数列。 第二行，需要插入的数字。输出排序后的数列。

样例输入

1 7 8 17 23 24 59 62 101

50

样例输出

1 7 8 17 23 24 50 59 62 101

分析：

1. 找到位置_____
2. 后挪_____
3. 插入_____

```
#include<iostream>
using namespace std;
int main(){
        int a[11],i,x,pos;
        for(i=1;i<=9;i++)
                cin>>a[i];
        cin>>x;//1 完成初始化
```

```
        for(i=1;i<=9;i++)//找到位置
                if(  1                    ){
                        break;
                }
        pos=  2          //2 记录找到的位置;
        for(i=   3                    9;i>=pos;i--)
                a[              ]=a[      ];//3 挪动
        a[          ]=       ;//4   插入
        for(i=1;i<=10;i++)cout<<a[i]<<endl;
        return 0;
}
```

3 [6855] 插队问题

有 n 个人（每个人有一个唯一的编号，用 1~n 之间的整数表示）在一个水龙头前排队准备接水，现在第 n 个人有特殊情况，经过协商，大家允许他插队到第 x 个位置。输出第 n 个人插队后的排队情况。

[输入格式]

第一行 1 个正整数 n，表示有 n 个人，2<n≤100。第二行包含 n 个正整数，之间用一个空格隔开，表示排在队伍中的第 1~ 第 n 个人的编号。第三行包含 1 个正整数 x，表示第 n 个人插队的位置，1≤x<n。

[输出格式] 一行包含 n 个正整数，表示第 n 个人插队后的排队情况。

[输入样例]

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

[输出样例]

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

问题分析：

n 个人的排队情况可以用数组 q 表示，q[i]表示排在第 i 个位置上的人。定义数组时多定义一个位置，**然后重复执行：q[i+1] = q[i]，其中，i 从 n ~ x。最后再执行 q[x] = q[n+1]，输出 q[1] ~ q[n]。**

```
    #include<iostream>
    using namespace std;
    int main(){
        int n,i,x,q[102];
        scanf( "%d " ,&n);
        for(i = 1; i <= n; i++)
            scanf( "%d " ,&q[i]);
        scanf( "%d" ,&x);
        for(i 1                    )
            q[_2      ] = q[       ];//移动
        q[3       ] = q[       ];//插入
        for(i = 1; i <= n; i++)
```

```
            printf( "%d  " ,q[i]);
        return 0;
    }
```

2.2 一维数组的删除

删除某一个元素，也需要先找到删除的位置(假设下标为 x)，将下标为 x+1 及其之后的所有元素依次向前移一位，覆盖原来位置上的元素，如图　所示。

图 2 元素删除示意图

删除元素的步骤：

如上图所示，要删除元素 h，假设 h=5,n=8,则

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

4 [6856] 队伍调整

有 n 个人（每个人有一个唯一的编号，用 1~n 之间的整数表示）在一个水龙头前排队准备接水，现在第 x 个人有特殊情况离开了队伍，求第 x 个人离开队伍后的排队情况。

第一行 1 个正整数 n，表示有 n 个人，2<n≤100。

第二行包含 n 个正整数,之间用一个空格隔开,表示排在队伍中的第 1 个到第 n 个人的编号。

第三行包含 1 个正整数 x，表示第 x 个人离开队伍，1≤x≤n。

输出一行包含 n-1 个正整数，之间用一个空格隔开，表示第 x 个人离开队伍后的排队情况。

[输入样例]

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

[输出样例]

```
7 2 4 5 6 1
```

```
#include<cstdio>
using namespace std;
int main(){
    int n,i,x,q[102];
    scanf( "%d" ,&n);
```

```
    for(i = 1; i <= n; i++) scanf( " %d " ,&q[i]);
    scanf( "%d" ,&x);
    for(i =_____1__________) q[_____2_____)] = q[___3_______];
    _____4_____
    for(i = 1; i < n; i++) printf( "%d  " ,q[i]);
    printf( " %d\n" ,q[n]);
    return 0;
}
```

5*[1468] 移动 n 个数

有 n 个整数(n<100)，使前面各数顺序向后移 m 个位置，最后 m 个数变成前面 m 个数。写一函数：实现以上功能，在主函数中输入 n 个数和输出调整后的 n 个数。（如果不是函数体，就不用函数）

输入

输入数据的个数 n 个整数 移动的位置 m

输出

移动后的 n 个数

样例输入

10

1 2 3 4 5 6 7 8 9 10

2

样例输出

9 10 1 2 3 4 5 6 7 8

方法一：生成新的数组

1. 如何把原数组 a 中的后 n-m 个元素赋值给 b 数组。

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[105],n,b[105],m;//  生成新数组新数组 b
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    cin>>m;
    for(int i=1;i<=n-m;i++)
        b[    ]=a[    ];//a 的前 n-m 的元素赋值到 b 的后面
    for(int i=1;i<=m;i++)
        b[    ] =a[    ] ;//a 的后面 m 个元素赋值给 b 的前面
    for(int t=1;t<=n;t++)
        cout<<b[t]<<" ";
    return 0;
```

```
}
```

方法 2：直接输出

```
#include<iostream>
using namespace std;
int main(){
    int n,m,a[101];
    cin>>n;
    for (int i=1;i<=n;i++){
        cin>>a[i];
    }
    cin>>m;
    for (int i=  __1_____________________){//
        cout<<a[i]<<" ";
    }
    for (int i=__2_____________________){
        cout<<a[i]<<" ";
    }
    return 0;
}
```

课堂作业列表

6855 插队问题　　1447 插入元素
6856 队伍调整　　3921 有序插入
1468 移动 n 个数　　2670 查找一个给定的值
1468 移动 n 个数　　6879 比身高
6858 整理题库

课前练习

```
int main()
{
    int x, y, t;
    cin >> x >> y;
    if (x < y)
    {
        t = x;x = y; y = t;
    }
    cout << x << " " << y;
    return 0;
}
```

输入：34 52
输出：_____52 34_____

```
int main()
{
    int i;
    int a[11];
    for (i = 1; i <= 10; i++)
        a[i] = i;
    for (i = 1; i <= 10; i++)
    {
        a[i] = a[i] + 1;
        cout << a[i] << " ";
    }
    return 0;
}
```

输出__2 3 4 5 6 7 8 9 10 11_

习题

1.有以下程序：

```
main( )
{  int  x[]={ 2,4,6,8,10 },t=1, i;
   for( i=1;i<x[1];i++ ) t=t*x[i];
   printf("%d",t);
}
```

程序运行后的输出结果是(C)

A)8　　B)384　　C)192　　D)1920

2.有以下程序：

```
main( )
{  int  a[3]={1},i,j;
   for( i=0;i<3;i++ )
       for( j=0;j<3;j++ )  a[i]=a[j]*3;
   printf("%d",a[3]);
}
```

程序运行后的输出结果是(A)

A)0　　B)不确定的值　　C)无输出　　D)3

3.有以下程序：

```
main( )
{  int  a[3]={1},i,j;
   for( i=0;i<3;i++ )
       for( j=0;j<3;j++ )  a[i]=a[j]*3;
   printf("%d",a[3]);
}
```

程序运行后的输出结果是( )

A)0　　B)不确定的值　　C)无输出　　D)3

4. 有以下程序：

```
main( )
{  int  i,a[5]={1,2,3,4,5}, t;
   t=a[0];
   for( i=1;i<5;i++ )  a[i-1]=a[i];
       a[i-1]=t;
   for( i=0;i<5;i++ ) printf("%4d",a[i]);}
```

程序运行后的输出结果是(D)

A)5 4 3 2 1　　B)2 3 4 1 1

C)5 4 3 2 2　　D)2 3 4 5 1

5.有以下程序：

```
main( )
{  int  a[3]={ 1,2,3 },i;
   for ( i=3; i>=1; i-- ) printf("%d  ",a[i]);
```

```
}
```

程序运行后的输出结果是(　　)B

A)1　2　3　　B)不确定的值　C)程序出错　D)3　2　1

6.以下程序运行后的输出结果是____。

1　2　0　6　0

```
#include <stdio.h>
int main( )
{    int  a[5]={1},i;
     for(i=1;i<5 ;i=i+2)  a[i]=2*i;
     for(i=0;i<5;i++)
printf("%d  ",a[i]);
}
```

7. 以下程序运行后的输出结果是____。

1　2　3　5　8　13　21　34

```
main( )
{    int  a[10],i;
     a[0]=0;  a[1]=1;
    for(i=2;i<10;i++)
      a[i]=a[i-1]+a[i-2];
    for(i=2; i<10; i++)
    printf("%d  ", a[i]);
}
```

8. n 个元素，把第 n 个数插入到第 x 位置上。

[输入样例]

7

7 2 3 4 5 6 1

3

[输出样例]

7 2 1 3 4 5 6

```
#include<cstdio>
using namespace std;
int main(){
    int n,i,x,q[102];
    scanf( "%d " ,&n);
    for(i = 1; i <= n; i++) scanf( "%d " ,&q[i]);
    __1__
    for( __2__ ) q[i+1] = q[i];
    q[x] = __3__;
    for(i = 1; i < n; i++) printf( "%d  " ,q[i]);
    printf( "%d\n " ,q[n]);
    return 0;
}
```

第四讲 状态变量

程序中的状态变量是指在程序设计过程中，描述某个对象形状、大小、颜色等属性以及事件状态的变量，通常用 0/1，true/false 等表示，比如用 0 表示红色、1 表示蓝色、2 表示白色。 理解状态变量是程序设计中非常重要的一环。

1 [2667] 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。

输入：每组输入数据的第一行有两个整数 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 每组输出包括一行，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 3 150 300 100 200 470 471	298

分析：初始所有[0,L]都有数，按要求把符合要求的数挪走，最后统计还剩下的数目。

1 如何表示初始状态都种着树？

定义个数组，用 a[i]表示树的状态，把所有的 a[i]都设置为 1，表示都种上树

_____

2 如何表示把区间[a,b]内的树挪走？

把 a[i]设置为 0，即用 1 表示有树，0 表示没有树。

_____

3. 如何统计马路上剩余的树的数目？

统计数组 a 中，有多少个 1。

_____

```
#include<iostream>
using namespace std;
int main(){
    int a[_1__________];
    int i,j,L,m,sum=0;
```

```
    int l1,l2;
    cin>>L>>m;
    for(i=0;i<=L;i++)a[i]=_2__________;
    for(i=1;i<=m;i++){
        cin>>l1>>l2;
        for(j=_3__________)a[_4___]=__5__________;
    }
    for(i=0;i<=L;i++)
        if(_6_______)
            _7_____________;
    cout<<sum<<endl;
    return 0;
}
```

2 [2948] 醉酒的狱卒

某个监狱有一排、共 n 间牢房，一间挨一间。每间牢房关着一名囚犯，每间牢房的门刚开始时都是关着的。有一天晚上，狱卒厌烦了看守工作，决定玩一个游戏。游戏的第 1 轮，他喝了一杯酒，然后沿着监狱，把所有牢房的门挨个挨个打开；游戏的第 2 轮，他又喝了一杯酒，然后沿着监狱，把编号为偶数的牢房的门关着；游戏的第 3 轮，他又喝了一杯酒，然后沿着监狱，对编号为 3 的倍数的牢房，如果牢房的门开着，则关上，否则打开；…，狱卒重复游戏 n 轮。游戏结束后，他喝下最后一杯酒，然后醉倒了。

这时，囚犯才意识到他们牢房的门可能是开着的，而且狱卒醉倒了，所以他们越狱了。

给定牢房的数目，求越狱囚犯的人数。

样例输入	样例输出
2 5 100	2 10

分析：

在本题中，n 轮游戏过后，哪些牢房的门是开着的，并无规律可循。但这个游戏的规则和过程都很简单：**游戏有 n 轮**，第 j 轮游戏是将编号为 j 的倍数的牢房状态变反，j = 1, 2, 3, …, n。这些规则和过程用程序能较容易地实现，所以适合采用"模拟"的思路求解。

具体实现时可以定义一个一维数组，表示每个牢房的状态，初始为 1，表示 Locked。模拟 n 轮游戏过程：第 j 轮时，改变牢房号为 j 的倍数的牢房的状态，为 0 改为 1，为 1 改为 0。最后统计状态为 0 的牢房数。

1 如何表示初始状态牢房的们都关着？

定义个数组，用 a[i]表示牢房的状态，把所有的 a[i]都设置为 1，表示牢房关着

_____

2 如何表示把当前的牢房的门取反操作？

如果 a[i]为 0，则变为 1；如果 a[i]为 1，则变为 0。也可以用取反操作“!”

_____

3.如何统计越狱囚犯的人数？

即门开着的数量，也就是 a[i]==0 的数量。

```
include<iostream>
using namespace std;
int main(){
    int a[101];//n 个牢房(编号从 1~n)的状态,1 表示锁着的
    int m,n,i,j,k,sum=0;
    cin>>m;////测试数据的个数,有 m 个监狱
    for(i=1;i<=m;i++){
        sum=0;
        cin>>n;//该监狱的牢房个数
        for(j=1;j<=n;j++)__1_____;//初始化牢房门关着
        for(j=1;j<=n;j++){//n 轮操作
            for(k=__2_____){//对 n 个牢房操作
                if(k%j==0)a[k]= __3_____
            }
        }
        for(j=1;j<=n;j++)
            if(__4_____ )sum++;
        cout<<sum<<endl;
    }
    return 0;
}
```

进一步优化：

```
for(j=1;j<=n;j++){//n 轮操作
    for(k=1;k<=n;k++){//对 n 个牢房操作
        if(k%j==0)a[k]=!a[k];
    }
}
```

可以不用遍历每一个牢房吗？

```
for(j=1;j<=n;j++){//n 轮操作
    for(k=j;k<=n;k=k+j){//对 n 个牢房操作
        a[k]=!a[k];
    }
}
```

3 [2669] 开关灯

假设有 N 盏灯(N 为不大于 5000 的正整数)，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(M 为不大于 N 的正整数)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，其间用逗号间隔。
输入 输入正整数 N 和 M，以单个空格隔开。
输出 顺次输出关闭的灯的编号，其间用逗号间隔。
样例输入 10 10
样例输出 1,4,9

思考：输出的时候如何实现输出的数据间有“,”隔开？

把输出分成两种状态考虑，第一个数和其它数。当处理第一数时就直接输出，当处理后面的数是输出“,数”。即定义一个变量 p，初始时 p=0，表示输出第一个数，输出第一个数后把 p 设置为 p=1,表示输出后面的数。

```
#include<iostream>
#include<cmath>
using namespace std;
int main(){
    int a[5005],n,m,i,j;
    cin>>n>>m;//n 灯 m 人
```

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

4 *[2667]去重

给定含有 n 个整数的序列，要求对这个序列进行去重操作。所谓去重，是指对这个序列中每个重复出现的数，只保留该数第一次出现的位置，删除其余位置。

输入包含两行：第一行包含一个正整数 n（1 ≤ n ≤ 20000），表示第二行序列中数字的个数；第二行包含 n 个整数，整数之间以一个空格分开。每个整数大于等于 10、小于等于 5000。

输出只有一行，按照输入的顺序输出其中不重复的数字，整数之间用一个空格分开。

样例输入

5

10 12 93 12 75

样例输出

10 12 93 75

常规方法：输入一个数，往前扫描这个是是否存在，如果存在就回退，删除该元素。

```
#include<iostream>
using namespace std;
 int main()
 {
       int a[2001];
       int n;
       scanf("%d",&n);
       for(int i=1;i<=n;i++)
       {
            scanf("%d",&a[i]);
           for(int j=1;j<i;j++)
            {
                if(a[j]==a[i])
                 {
                     i=i-1;
                     n=n-1;
                 }
            }
       }
       for(int i=1;i<=n;i++)
           printf("%d ",a[i]);
 }
```

用状态数组标记该数是否已经出现过的数

```
#include<iostream>
using namespace std;

int main()
```

```
{
    int n;
    int a[20005], b[5005];
    cin>>n;
    for(int i=1;i<=n;i++)
        cin>>a[i];
    for(int i=0;i<=5000;i++)
        b[i]= __1__0;//初始数组 b,用来标记该数是否已经出现过
    for(int i=1;i<=n;i++)
    {
        if(__2__ b[a[i]]==0)
        {
            cout<<a[i]<<" ";
            __3__ b[a[i]]=1; //这句代码的含义是？
        }
    }
    return 0;
}
```

5*[1609] 素数-大数据

素数是这样的整数，它除了能表示为它自己和 1 的乘积以外，不能表示为任何其它两个整数的乘积。例如，15＝3*5，所以 15 不是素数。又如，12＝6*2＝4*3，所以 12 也不是素数。另一方面，13 除了等于 13*1 以外，不能表示为其它任何两个整数的乘积，所以 13 是一个素数。 你的任务是计算出所有小于等于给定正整数的素数个数(我们认为 1 不是素数)。

输入输入文件中包含多个测试数据。每个测试数据占一行，为一个正整数 N(0<N≤10000000)，即给定正整数。测试数据一直到文件尾。

输出对输入文件中的每个正整数，输出所有小于等于给定它的素数个数。

样例输入	样例输出
2	1
3	2
4	2
5	3
9	4

课前练习

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
```

```
#include <iostream>
using namespace std;
int main()
{
```

```
    int x[10]={ 77,34,85,74,12,21,64,90,101,9 };
    int i,m;
    m=0;
    for ( i=1;i<10;i++ )
    if( x[i]>x[m] ) m=i;
    cout<<m;
    return 0;
}
```

结果是_____8_____

```
    int i, j, k, t;
    int a[8];
    for (i = 1; i <= 7; i++)a[i] = 0;
    for (i = 1; i <= 4; i++)a[i] = i;
    t = a[7];
    for (i = 7; i >= 2; i--)
        a[i] = a[i - 1];
    a[1] = t;
    for (i = 1; i <= 7; i++)
        cout << a[i];
    return 0;
}
```

结果是____0123400_____

习题

1. 以下程序的输出结果是(A)

```
main( )
{ int a[10]={3,8,5,6,4,3,0,12,1,2};
  int i,j=0;
  for(i=0; i<10; i++)
     if(a[i]%2==0) a[j++]=a[i];
  for(i=0; i<j; i++) printf("%d,",a[i]);
  printf("\n");
}
```

A)8,6,4,0,12,2　　B)6,4,0,12,2　C)8,6,4,0,12　D)6,4,0,12

2.有以下程序：

```
main( )
{  int  w[10]={1,2,3,4,5,6,7,8,9,10},i,t;
   for( i=0;i<10;i++ )
   {   t=w[i];
       w[i]=w[i]*w[i]%10;
       if( w[i]==t ) break;
}
printf("%d",w[5]);
}
```

程序运行后的输出结果是(B)

A)1　　B)6　　C)5　　D)2

3.用数组存储从键盘上输入的 10 个整数，并求这 10 个整数的最大值、最小值和平均值。

```
#include<iostream>
using namespace std;
int main( )
```

```
{
    int a[10], max, min;
    double average;
    for( i=0; i<10; i++ )
        scanf( "%d", &a[i] );
    __1___________
    average = a[0];
    for( i=1; i<10; i++ )
    {
        if( a[i]>max )  max =__2_____
        if( a[i]<min )  min = a[i];
        average +=___3_____ ;
    }
    average = ___4_________
    printf( "max = %d, min = %d, average = %.1f\n", max, min, average );
}
```

4 以下程序运行后的输出结果是____。

```
#include<iostream>
using namespace std;
int main(){
  int a[10005];
  int n,x;
  cin>>n;
  for(int i=1;i<=n;i++)
    cin>>a[i];
  cin>>x;
  int flag=0;
  for(int i=1;i<=n;i++)
  if(a[i]==x){
      cout<<i<<endl;
      break;
  }
  return 0;
}
```

输入

5

2 3 6 7 3

3

输出 ___2_____

5 以下程序运行后的输出结果是____。

```
#include <iostream>
using namespace std;
int main()
{
    int n, i, k;
    int sum, ans, min;
    cin >> n;
    sum = 0,min = 0,ans = -32764;
    for (i = 1; i <= n; i++)
    {
        cin >> k;
        sum = sum + k;
        if (sum - min > ans)
            ans = sum - min;
        if (sum < min)
            min = sum;
    }
    cout << ans << endl;
    return 0;
}
```

输入：

6

5 -1 0 3 -5 9

输出：___11_____

第五讲 二维数组

重点
1. 理解数组的含义。
2. 学会二维数组的定义。
3. 掌握二维数组的元素引用和物理存储方式。

当一维数组元素的类型也是一维数组时，便构成了“数组的数组”，即二维数组。

5.1 二维数组的定义

二维数组类型定义的一般形式是：

类型符 数组名[常量表达式 1][常量表达式 2];

其中，<常量表达式 1>表示第一维下标的长度，<常量表达式 2>表示第二维下标的长度。二维数组的大小（元素个数）为：常量表达式 1×常量表达式 2。
例如，int a[3][4]; 定义了一个 3 行 4 列的数组，数组名为 a，其元素变量的类型为整型。该数组的元素共有 3×4 个，其元素为：

a[0][0], a[0][1], a[0][2], a[0][3]
a[1][0], a[1][1], a[1][2], a[1][3]
a[2][0], a[2][1], a[2][2], a[2][3]

二维数组 a[3][4]理解为：有 3 个元素 a[0]、a[1]、a[2]。a[0]、a[1]、a[2]又当作一维数组名，每个数组名各包含 4 个元素，如图所示。

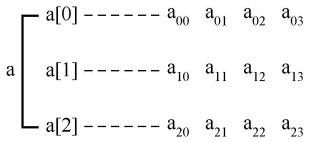


图 5-1　二维数组可理解为由多个一维数组构成

5.2　二维数组元素的引用

二维数组的元素也称为双下标变量，其表示的形式为：
数组名[下标 1][下标 2]
其中，下标应为整型常量或整型表达式。例如，
a[1][2]=10;　　//将第 1 行第 2 列元素赋值为 10
a[i][j]＝a[i-1][j-1]+2;
在程序中常常通过二重循环来使用二维数组元素。

5.3　二维数组的初始化

与一维数组一样，也可以对二维数组进行初始化。在对二维数组进行初始化时要注意以下几点。

1）二维数组可按行分段赋值，也可按行连续赋值。例如，

```
int a[5][3]={ {80,75,92},{61,65,72},{59,63,70},{85,87,90}, {76,97,85} };
int a[5][3]={ 80,75,92,61,65,72,59,63,70,85,87,90,76,97,85 };
```

这两种赋初值的结果是完全相同的。

2）可以只对部分元素赋初值，未赋初值的元素自动取 0 值。例如，int a[3][3]={{1},{0,2},{0,0,3}}; 赋值后的数组为：

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 2 & 0\\ 0 & 0 & 3\end{bmatrix}$$

3）如对全部元素赋初值，则第一维的长度可以不给出（但一对方括号不能省略）。例如，int a[3][3]={1,2,3,4,5,6,7,8,9}; 可以写为 int a[][3]={1,2,3,4,5,6,7,8,9};，赋值后的数组为：

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 9\end{bmatrix}$$

说明：如果省略第一维的长度按行连续赋初值，系统根据数据的多少自动确定数组的行数。例如，int a[][3]={1,2,3,4,5,6,7};，由于每行 3 个元素，而初值有 7 个，所以赋值后的数组为：

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 0 & 0\end{bmatrix}$$

4）给二维数组输入数据

设所有变量及数组已定义，其程序段如下。

```
for(i=0;i<N;i++)              // N 为第一维下标的长度
   for(j=0;j<M;j++)           // M 为第二维下标的长度
      cin>>a[i][j];           //scanf("%d",&a[i][j]);
```

5.4 二维数组程序举例

1 [2115] 种萝卜

题目描述

一只大白兔准备在一个 n*n 的农田种萝卜，因为大白兔比较懒，所以他想用已有的坑来种萝卜，现在给你 n*n 的农田，让你告诉他能种多少萝卜。

输入一个 n,然后给你一个 n*n(n<=100)的矩阵，每个单元只包含一个数字 0 或 1,0 表示坑，在坑内可以种萝卜，1 表示平滑的土地,不能种萝卜。
输出他最多能种多少个萝卜。
输入：输入一个 n,然后给你一个 n*n(n<=100)的矩阵，每个单元只包含一个数字 0 或 1,0 表示坑，在坑内可以种萝卜，1 表示平滑的土地,不能种萝卜。
输出
输出他最多能种多少个萝卜。
样例输入
3
1 1 1
1 0 0
1 0 1
样例输出
3

```
#include<iostream>
using namespace std;
int main()
{
        int a[105][105],n,k=0,i;
        cin>>n;
        for(int i=1;i<=n;i++)
          for(int j=1;j<=n;j++){
                        cin>>a[i][j];
                    if(a[i][j]==0)
                          k++;
        }
        cout<<k;
}
```

2 [3291] 求最大值

有一个 3×4 的矩阵，要求编程序求出其中值最大的那个元素的值，以及其所在的行号和列号。

$$\begin{bmatrix} 5 & 12 & 23 & 56 \\ 19 & 28 & 37 & 46 \\ -12 & -34 & 6 & 8 \end{bmatrix}$$

分析：求一组数中的最大数，要用到“摆擂台”的思想。开始时把 a[0][0]的值赋给变量 max，然后让每一个元素与它比较。如果这个元素比 max 的当前值还要大，则把该元素的值赋值给 max，并且用 row 和 column 记录该元素的两个下标。这样 max 就是当前找到的最大的数。一直比较到最后一个元素比完为止。max 最后的值就是数组所有元素中的最大值。

```
#include <iostream>
using namespace std;
int main( )
```

```
{
    int i, j, row, column, max;
    int a[3][4];
    for(i=0;i<3;i++)
        for(__1__________________)
        cin>>_2_____________;
    ___3_____________________;  //初始化
    for( i=0; i<=2; i++ )        //从第 0 行~第 2 行
    {
        for( j=0; j<=3; j++ )    //从第 0 列~第 3 列
        {
            if( __4_____________ )  //如果某元素大于当前的 max
            {
                max =_5_____;  //max 将取该元素的值
                row = _6____;
                column = _7_____;//记下该元素的行号 i 和列号 j
            }
        }
    }
    printf( "max=%d, row=%d, column=%d\n", max, row+1, column+1 );
    return 0;
}
```

3 [7613] 生成一个矩阵并输出

定义 1 个 m*n 的二维数组 a，数组元素的值由下式给出，按矩阵的形式输出 a。
a[i][j] = i + j　　(1 ≤ i ≤ m, 1 ≤ j ≤ n ,m<=20,n<=20)

输入 m,n 输出 按矩阵的形式输出 a。
样例输入 2 3
样例输出
2 3 4
3 4 5

分析：**数组中的每个元素不是自己直接输入，而是在程序中生成，理解生成规则。**

```
#include <iostream>
using namespace std;
int main( )
{    int i, j,m,n;
     int a[30][30];
     cin>>m>>n;
```

```
    for ( i = 1; i <= m; i++ ){
        for ( j = 1; j<=n; j++ )
            printf ( "%d ", a[i][j] );
        printf ( "\n" );
    }
    return 0;
}
```

4 [1452] 二维数组转置

写一个程序，使给定的一个二维数组（3 × 3）转置，即行列互换。

输入一个 3x3 的矩阵输出转置后的矩阵

样例输入	样例输出
1 2 3 4 5 6 7 8 9	1 4 7 2 5 8 3 6 9

思考:转换的规律是什么？

a[1][2]->a[2][1]

a[1][3]->a[3][1]

a[2][1]->a[1][2]

....

a[ ][ ]=a[ ][ ]

```
#include<iostream>
using namespace std;
int main(){
    int a[4][4];
    for(int i=1;i<=3;i++)
        for(int j=1;j<=3;j++)
            cin>>a[i][j];
```

```
return 0;
}
```

5 *[2046] 杨辉三角

还记得中学时候学过的杨辉三角吗？具体的定义这里不再描述，你可以参考以下的图形：

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
```

输入

每个测试实例的输入只包含一个正整数 n（1<=n<=30），表示将要输出的杨辉三角的层数。

输出

对应于每一个输入，请输出相应层数的杨辉三角。

样例输入 3

样例输出

```
1
1 1
1 2 1
```

分析：该三角形的第一列和对角线都是 1，同时从第三行开始，每个元素的值，都是正上方的元素的值与左侧上方的元素值之和。假设当前元素为 a[i][j],则

```
a[i][j]=_____________________________
```

初始化边界：

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

```
#include<iostream>
using namespace std;
int main()
{
    int a[30][30],i,j,n;
    cin>>n;
    for(i=1;i<=n;i++) {
        _1__________________
    }
    for(i=3;_2________________)
        for(_3________________)
            _4________________
```

```
    for(i=1;i<=n;i++) {
        for(j=1;j<=i;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
   return 0;
}
```

如果下标从 0 开始呢？

```
#include<iostream>
using namespace std;
int main()
{
    int a[30][30],i,j,n;
    cin>>n;
    for(i=0;i<n;i++) {
        a[i][i]=1;a[_1___][____]=1;
    }
    for(i=2;i<n;i++)
        for(j=_2_____________;j++)
            a[i][j]=_3_____________;
    for(i=0;i<n;i++) {
        for(j=__4_____________;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
   return 0;
}
```

课堂作业列表

2682 计算鞍点　　3291 求矩阵最大元素
2046 杨辉三角　　2688 图像模糊处理
2685 矩阵乘法　　2240 对角线求和
1452 转置

课前练习

#include <iostream>
using namespace std;
int main()
{
    int s, x;
	#include<iostream>
	using namespace std;
	int main( )
	{ int w[10]={1,2,3,4,5,6,7,8,9,10}
	    int i,t;

```
    x = 0; s = 0;
    while (s < 55)
    {
        x = x + 1;
        s = s + x;
    }
    cout << x << endl;
    return 0;
}
```

输出：_____10_____

```
   for( i=0;i<10;i++ )
   {   t=w[i];
       w[i]=w[i]*w[i]%10;
       if( w[i]==t ) break;
    }
    printf("%d",w[5]);
}
```

结果是 _____6_____

习题

1. 以下数组定义中错误的是(　)

A) int x[][3]={0};　　B) int x[2][3]={{1,2},{3,4},{5,6}};

C) nt x[][3]={{1,2,3},{4,5,6}};　D)int x[2][3]={1,2,3,4,5,6};

2. 数组定义为 int a[3][2]={1,2,3,4,5,6}，值为6的数组元素是(　　)

a[3][2]　　B. a[2][1]　　C. a[1][2]　　D. a[2][3]

3.对二维数组a的正确初始化形式是(　　)

A. int a[2][3]={ 1,2,3,4,5,6,7 };

B. int a[][3]={{1,2,3}, {4,5,6}};

C. int a[2][3]={{1,2},{3,4},{5,6}};

D. int a[2][3]={{},{4,5,6}};

4. 若有定义语句：int a[3][6]; 按在内存中的存放顺序，a数组的第10个元素是(　　)

A）a[0][4]　　B)a[1][3]　　C)a[0][3]　　D)a[1][4]

5. 合法的数组定义是(　)。

A. int a[3][ ]={0,1,2,3,4,5};　　B. int a[2][3] ={{0,1,2},{3,4}};

C. int a[2][3]={0,1,2,3,4,5,6};　D. int a[2][3]={{0},{1,2},{4,5}};

6. 若有说明"int s[2][5]，n=3;",则对s数组元素的非法引用是(　　)

A)s[1][2+3]　　B)s[0][n+1]　　C)s[1][4-2]　　D)s[n-2][0]

7.以下程序运行后的输出结果是__1 2 0 6 0__

```
int main( )
{    int  a[5]={1},i;
     for(i=1;i<5 ;i=i+2)  a[i]=2*i;
     for(i=0;i<5;i++)  printf("%d  ",a[i]);
}
```

8.以下程序运行后的输出结果是 __1 2 3 5 8 13 21 34__

```
main( )
{    int a[10],i;
     a[0]=0,a[1]=1;
```

```
      for(i=2; i<10; i++) a[i]=a[i-1]+a[i-2];
      for(i=2; i<10; i++)  printf("%d  ", a[i]);
}
```

9.下面程序的输出结果是：

```
int main()
{
 int a[3][3]={0,1,2,3,4,5,6,7,8},i,j,t;
 for(i=0;i<3;i++)
  for(j=0;j<3;j++)
   a[i][j]+=a[i][0];
 for(i=0;i<3;i++)
  printf(""%d,"",a[i][i]);}
```

运行结果_____01020_____

10.已知整数数组 b[2][5]={{7,15,2,8,20}，{12,25,37,16,28}}，求数组中所有元素的最大值，请填空。

```
#include<iostream>
using namespace std;
int main()
{
    int b[2][5]=_____1____;
    int i,j,c,d,k=0;
    for(i=0;i<2;i++)
    {
        for(j=0;_____2__;j++)
        {
            _____3_;
            c=i,d=j;
        }
    }
    cout<<"b["<<c<<"]["<<d<<"]="<<k<<endl;
    return 0;
}
```

第六讲 理解矩阵

重点

1. 掌握数组的坐标与表达。
2. 掌握二维数组常见应用。

在数学中，矩阵（Matrix）是一个按照长方阵列排列的实数集合，最早来自于方程组的系数及常数所构成的方阵。这一概念由 19 世纪英国数学家凯利首先提出。作为解决线性方程的工具，矩阵也有不短的历史。最早在东汉前期的《九章算术》中，用分离系数法表示线性方程组，得到了系数矩阵。

矩阵的概念在 19 世纪逐渐形成。1800 年代，高斯和威廉·若尔当建立了高斯—若尔当消去法。1844 年，德国数学家费迪南·艾森斯坦（F.Eisenstein）讨论了“变换”（矩阵）及其乘积。1850 年，英国数学家詹姆斯·约瑟夫·西尔维斯特（James Joseph Sylvester）首先使用矩阵一词。

英国数学家阿瑟·凯利被公认为矩阵论的奠基人，他开始将矩阵作为独立的数学对象研究。

6.1 理解位置

思考：当前位置是(i, j)，则包围(i, j)的 8 个点如何表示？

i-1, j-1		
	i, j	
	i+1, j	

6.2 矩阵操作

1[3155] 构建矩阵

现请你构建一个 N*N 的矩阵，第 i 行 j 列的元素为 i 与 j 的乘积。（i，j 均从 1 开始）

输入的第一行为一个正整数 C，表示测试样例的个数。然后是 C 行测试样例，每行为一个整数 N（1<=N<=9），表示矩阵的行列数。

输出对于每一组输入，输出构建的矩阵。

样例输入	样例输出
2 1 4	1 1 2 3 4 2 4 6 8 3 6 9 12 4 8 12 16

分析：**数组中的每个元素不是自己直接输入，而是在程序中生成，理解生成规则。**
第 i 行 j 列的元素为 i 与 j 的乘积:a[i][j]=_____

```
#include<iostream>
using namespace std;
int main(){
        int c,n;
        int a[10][10],i,j;//i 是行， j 是列
        cin>>c;
        for(int k=1;k<=c;k++)
        {
                cin>>n;
                for(i=1;i<=n;i++)
                {
                        for(j=1;j<=n;j++)
                                cout<<i*j<<' ';
                        cout<<endl;
                }
        }
return 0;
}
```

2 [2243] 判断上三角矩阵

输入一个正整数 n(1<=n<=6)和 n 阶矩阵 a 中的元素，如果 a 是上三角矩阵，输出“YES”；否则，输出“NO”（上三角矩阵，即**主对角线以下的元素都为 0**，主对角线为从矩阵的左上角至右下角的连线）。

输入 2 1 0 -8 2	输出 NO
样例输入 3 1 2 3 0 4 5 0 0 6	样例输出 YES

思考：如何表示为值的元素 0 的区域？

```
 for(i= _____i=2;i<=n;i++){
        for(j= _____j=1;j<i;j++){
            a[i][j]都等于 0
        }
 }
#include<iostream>
```

```
using namespace std;
int main(){
    int n,a[10][10],i,j,flag=1;
    cin>>n;
    for(i=1;i<=n;i++){
        for(j=1;j<=n;j++){
            cin>>a[i][j];
        }
    }
```

```
    if(flag==1)cout<<"YES";
    else cout<<"NO";
    return 0;
}
```

3 [2679] 矩阵交换行

给定一个 5×5 的矩阵(数学上，一个 r×c 的矩阵是一个由 r 行 c 列元素排列成的矩形阵列)，将第 n 行和第 m 行交换，输出交换后的结果。

输入　输入共 6 行，前 5 行为矩阵的每一行元素,元素与元素之间以一个空格分开。第 6 行包含两个整数 m、n，以一个空格分开（1 ≤ m,n ≤ 5）。
输出　输出交换之后的矩阵，矩阵的每一行元素占一行，元素之间以一个空格分开。
样例输入
1 2 2 1 2
5 6 7 8 3
9 3 0 5 3
7 2 1 4 6
3 0 8 2 4
1 5
样例输出
3 0 8 2 4
5 6 7 8 3
9 3 0 5 3
7 2 1 4 6
1 2 2 1 2

分析：找规律，如果第 i 行和第 j 行的值发生的交换则

t=a[ ][ ],a[ ][ ]=a[ ][ ],a[ ][ ]=t;

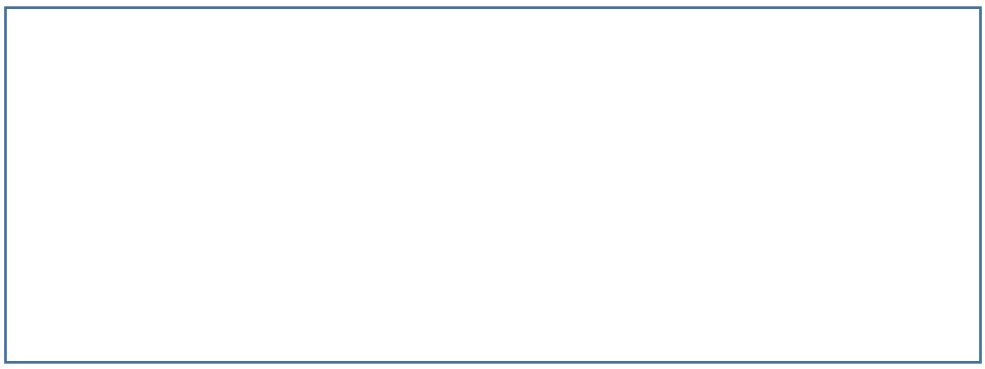
```
#include<iostream>
using namespace std;
int main()
{
    int n,m, mp[9][9];
    for(int i=1;i<=5;i++)
        for(int j=1;j<=5;j++)
            cin>>mp[i][j];
    cin>>n>>m;
```

```
}
```

4 [2681] 矩阵边缘元素之和

输入一个整数矩阵，计算位于矩阵边缘的元素之和。所谓矩阵边缘的元素，就是第一行和最后一行的元素以及第一列和最后一列的元素。

```
#include<iostream>
using namespace std;
int main(){
        int a[100][100],m,n,sum=0;
        int i,j;
        cin>>m>>n;
```



```
      cout<<sum;
        return 0;
}
```

5 *[2680] 同行列对角线的格

输入三个自然数 N，i，j（1≤i≤n，1≤j≤n），输出在一个 N*N 格的棋盘中(行列均从 1 开始编号)，与**格子(i，j)同行、同列、同一对角线的所有格子的位置**。

如：n=4，i=2，j=3 表示了棋盘中的第二行第三列的格子，

当 n=4，i=2，j=3 时，输出的结果是：

(2,1) (2,2) (2,3) (2,4)　　同一行上格子的位置

(1,3) (2,3) (3,3) (4,3)　　同一列上格子的位置

(1,2) (2,3) (3,4)　　左上到右下对角线上的格子的位置

(4,1) (3,2) (2,3) (1,4)　　左下到右上对角线上的格子的位置

输入

一行，三个自然数 N，i，j，相邻两个数之间用单个空格隔开(1≤N≤10)。

输出

第一行：从左到右输出同一行格子位置；

第二行：从上到下输出同一列格子位置；

第三行：从左上到右下输出同一对角线格子位置；

第四行：从左下到右上输出同一对角线格子位置。

其中每个格子位置用如下格式输出：(x,y)，x 为行号，y 为列号，采用英文标点，中间无空格。相邻两个格子位置之间用单个空格隔开。

样例输入

4 2 3

样例输出

(2,1) (2,2) (2,3) (2,4)

(1,3) (2,3) (3,3) (4,3)

(1,2) (2,3) (3,4)

(4,1) (3,2) (2,3) (1,4)

	1 2	1 3	1 4
2 1	2 2	**2,3**	2 4
	3 2	3 3	3 4
4 1		4 3	

分析：

1. 行相同的元素具有什么特点：_____

2. 列相同的元素具有什么特点：_____

3. 左上-> 右下 对角线 相同的元素具有什么特点：_____

4. 左下-> 右上 对角线 相同的元素具有什么特点：_____

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
        int n,x,y,i,j;
        scanf("%d%d%d",&n,&x,&y);
        for(i=1;i<=n;i++)//输出行
                printf("(%d,%d) ",x,i);
        printf("\n");
```

```
        for(i=1;i<=n;i++)//输出列
                printf("(%d,%d) ",i,y);
        printf("\n");
        for(i=1;i<=n;i++)
                for(j=1;j<=n;j++)//(1,2) (2,3) (3,4)
                        if(x-y==i-j)//输出对角线 规律：2-1=3-2=4-3
                                printf("(%d,%d) ",i,j);
        printf("\n");
        for(i=n;i>=1;i--)
                for(j=n;j>=1;j--)//(4,1) (3,2) (2,3) (1,4)
                        if(x+y==i+j)//输出斜对角线 规律：4+1=3+2=2+3=1+4
                                printf("(%d,%d) ",i,j);
        printf("\n");
        return 0;
}
```

写法二：

```
#include <iostream>
using namespace std;
int main(){
        int n,i,j,p,q;
        cin>>n>>i>>j;
        for(p=1;p<=n;p++)//输出行 (i,1) (i,2)...
                cout<<"("<<i<<","<<p<<")"<<" ";
        cout<<endl;

        for(p=1;p<=n;p++)//输出列 (1,j) (2,j)...
                cout<<"("<<p<<","<<j<<")"<<" ";
        cout<<endl;
        int x=i,y=j;
        while(x&&y){//找到左上角    x!=0&&y!=0
                x--;y--;
        }
        x++,y++;//退出上面的循环是 多减了一次，加回去
        while(x<=n&y<=n){//输出对角线上的元素
                cout<<"("<<x<<","<<y<<")"<<" ";
                x++,y++;
        }
        cout<<endl;
        x=i,y=j;
        while(x<=n&&y>=1){//找到左下角
                x++,y--;
        }
```

```
        x--,y++;
        while(x&&y<=n){//
                cout<<"("<<x<<","<<y<<")"<<" ";
                x--,y++;
        }
return 0;
}
```

6*[3753]蛇形填数

在 n*n 方阵里填入 1,2,3,…,n*n，要求填成蛇形。例如 n=4 时方阵为：

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

上面的方阵中，多余的空格只是为了便于观察规律，不必严格输出，n<=8。

输入

一个整数 n，n≤8。

输出

n 行，每行 n 个整数，用空格分隔。

样例输入

4

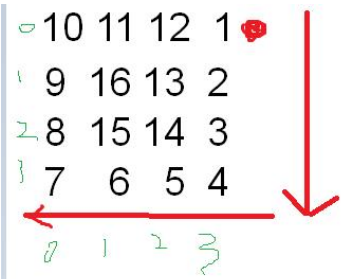
样例输出

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

分析：

先判断后移动，从右上到右下，再从右下到左下，再从左下到左上，再从左上到右上，然后再这样循环下去。我们可以用一个二维数组来储存题目中的方阵，声明一个 int a[MAXN][MAXN]。

从 1 开始依次填写。设“笔”的坐标为(x,y)，则一开始 x=0，y=n-1，即第 0 行，第 n-1 列（别忘了行列的范围是 0 到 n-1，没有第 n 列）。“笔”的移动轨迹是：下，下，下，左，左，左，上，上，上，右，右，下，下，左，上。总之，先是下，到不能填了为止，然后是左，接着是上，最后是右。“不能填”是指再走就出界（例如 4→5），或者再走就要走到以前填过的格子（例如 12→13）。如果我们把所有格子初始为 0，就能很方便地加以判断。



tot=a[x=0][y=n-1]=1;
a[++x][y]=++tot;
a[x][--y]=++tot;
a[--x][y]=++tot;
a[x][++y]=++tot;

下边界的判断：x+1<n
左边界的判断：y-1>0
上边界的判断：x-1>=0
右边界的判断：y+1<n

```
#include<iostream>//
#include<cstring>
using namespace std;
int a[100][100];

int main(){
        int x,y,n,total;
        cin>>n;
        memset(a,0,sizeof(a));
        x=0,y=n-1;
        a[x][y]=total=1; //初始状态，从右上角开始
        while(total<n*n){
                while(x+1<n&&!a[x+1][y])
                        a[++x][y]=++total;//x++; total++;a[x][y]=total;
                while(y-1>=0&&!a[x][y-1])
                        a[x][--y]=++total;
                while(x-1>=0&&!a[x-1][y])
                        a[--x][y]=++total;
                while(y+1<n&&!a[x][y+1])
                        a[x][++y]=++total;
        }
        for(x=0;x<n;x++){
                for(y=0;y<n;y++)
                        printf("%d ",a[x][y]);
                printf("\n");
        }
        return 0;
}
```

```
#include<stdio.h>
int main(){
        int a,b,c,d,n,sum=1;
        int yi[101][101];
        scanf("%d",&n);
        for(a=0;a<=(n-1)/2;a++){
                for(b=a;b<=n-a-1;b++)
                        yi[b][n-a-1]=sum++;
                for(b=n-2-a;b>=a;b--)
                        yi[n-a-1][b]=sum++;
                for(b=n-a-2;b>=a;b--)
                        yi[b][a]=sum++;
                for(b=a+1;b<n-a-1;b++)
                        yi[a][b]=sum++;
        }
        for(c=0;c<n;c++){
                for(d=0;d<n;d++)
                printf("%d ",yi[c][d]);
                printf("\n");
        }
}
```

7 *[3356] 1234 方阵（phalanx）

编程打印如下规律的 n*n 方阵。输入 n，按规律输出方阵。
方阵规律如下图：使左对角线和右对角线上的元素为 0,它们上方的元素为 1,左方的元素为 2,下方元素为 3,右方元素为 4,下图是一个符合条件的 5 阶矩阵。

```
0 1 1 1 0
2 0 1 0 4
2 2 0 4 4
2 0 3 0 4
0 3 3 3 0
```

输入
正整数 n（<=100）。
输出
所需的方阵。
样例输入
5
样例输出
0 1 1 1 0
2 0 1 0 4
2 2 0 4 4
2 0 3 0 4

03330

```
#include<cstdio>
#include<iostream>
using namespace std;
int a[105][105]={0};
int main()
{
        int n,i,j;
        cin>>n;
        for (i=1;i<=n;i=i+1)
                for (j=1;j<=n;++j)
                {
                        if (i+j>n+1&&i>j) a[i][j]=3;
                        if (i+j>n+1&&i<j) a[i][j]=4;
                        if (i+j<n+1&&i<j) a[i][j]=1;
                        if (i+j<n+1&&i>j) a[i][j]=2;
                }
        for (i=1;i<=n;i=i+1)
                for (j=1;j<=n;j++)
                        if (j==n) cout<<a[i][j]<<endl;
                                else cout<<a[i][j]<<" ";
        return 0;
}
```

课前练习

1. 以下程序的运行结果是 ___40_____

```
#include <iostream>
using namespace std;
const int n=5;
int main()
{
    int a[n][n]={{3,2,4,1,5},
                 {8,7,2,5,6},
                 {6,9,1,4,3},
                 {5,5,3,6,2},
                 {2,8,1,8,6}};
    int s=0;
    for(int i=0;i<n;i++)
        s+=a[i][i]+a[i][n-i-1];
    if(n%2==1)
```

2 以下程序的运行结果是

_____

```
    1
    1    1
    1    2    1
    1    3    3    1
    1    4    6    4    1
```

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    const int m=5;
    int a[m][m];
    for(int i=0;i<m;i++)
```

```
        s-=a[n/2][n/2];
    cout<<s<<endl;
    return 0;
}
```

```
    {
        a[i][0]=1;
        a[i][i]=1;
        for(int j=1;j<i;j++)
        a[i][j]=a[i-1][j-1]+a[i-1][j];
    }
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<=i;j++)
                cout<<setw(5)<<a[i][j];
        cout<<endl;
    }
}
```

习题

1.有以下程序：

```
main( )
{
    int  x[4][4]={{1,2,3},{3,4,5},{4,5,6},{4,5,6,7}},i,j,t=0;
    for(i=0;i<4;i++ )
    for( j=i;j<4;j++ ) t=t+x[i][j];
    printf("%d\n",t);
}
```

程序运行后的输出结果是(A   )

A)28　　B)45　　C)18　　D)55

2.有以下程序：

```
main( )
{
    int  arr[3][3],i,j;
    for( i=1;i<3;i++ )
       for( j=1;j<3;j++ ) arr[i][j]=i%j+j%i;
    for( i=1;i<3;i++ ){
        for( j=1;j<3;j++ ) printf("%3d",arr[i][j]);
        printf("\n");
    }
}
```

程序运行后的输出结果是(D   )

A)1 0　　B)0 0　　C)1 1　　D)0 1

  0 1　　 1 1　　 0 0　　 1 0

3.有以下程序：

```
main( )
{  int  x[4][4]={{1,2,3},{3,4,5},{4,5,6},{4,5,6,7}},i,j,t=0;
```

```
    for(i=0;i<4;i++ )
        for( j=i;j<4;j++ ) t=t+x[i][j];
    printf("%d\n",t);
}
```

程序运行后的输出结果是(A)

A)28　　B)45　　C)18　　D)55

4.读程序写结果_____

```
#include <iostream>
using namespace std;
int main()
{
    int a[3][3]={0},i,j;
    for(i=0;i<3;i++)
    {   for(j=0;j<3;j++)
        {   if(i==1)
                a[i][j]=a[i-1][a[i-1][j]]+1;
            else a[i][j]=j;
            printf("%4d",a[i][j]);
        }
        printf("\n");
    }
}
```

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

5. 下面程序的功能是计算矩阵 arr 主对角线上的元素之和。请填空。

```
main()
{   int arr[5][5],i,j,total;
    _____  ;
    for(i=0;i<5;j++)
        for(j=0;j<5;j++)_____;     /*输入数据*/
    for(i=0;i<5;i++)
        for(j=0;i<5;j++)
            if( _____) total=total+arr[i][j];
    printf(“The total is %d\n”,total);
}
```

6 阅读程序

```
#include <iostream>
using namespace std;
int main()
{
    int a[9][9];
    int i, j, sum;
    for (i = 1; i <= 8; i++)
```

```
        for (j = 1; j <= 8; j++)
            cin >> a[i][j];
    sum = 0;
    for (i = 1; i <= 8; i++)
        for (j = 1; j <= 8; j++)
            if (i == j || (i + j == 9))
                sum = sum + a[i][j];
    cout << sum << endl;
    return 0;
}
```

输入：

```
2 3 4 5 6 1 0 9
4 5 6 2 1 4 9 4
9 8 7 6 4 5 2 3
5 6 7 8 2 1 2 3
8 9 0 0 3 4 5 2
9 8 5 6 7 8 9 0
2 3 2 2 2 1 2 3
6 6 6 6 6 6 6 6
```

输出：80

第七讲 图像处理

重点

1. 强化二维数组的理解。
2. 理解图像的像素和基本操作。

图像处理(image processing)是指用计算机对图像进行分析，以达到所需结果的技术。图像处理的对象就是指数字图像，数字图像是指用工业相机、摄像机、扫描仪等设备经过拍摄得到的一个大的二维数组，该数组的元素称为像素，其值称为灰度值。图像处理技术一般包括图像压缩，增强和复原，匹配、描述和识别等几个部分。

常用的图像处理软件有 photoshop、画图等，其处理的内容包括对图像进行旋转、模糊处理等。

1 [2687] 图像旋转

输入一个 n 行 m 列的黑白图像，将它顺时针旋转 90 度后输出。

样例输入	样例输出
3 3 1 2 3 4 5 6 7 8 9	7 4 1 8 5 2 9 6 3

思考:转换的规律是什么？

a[1][1]->a[1][3]　　[2][1]->a[1][2]

a[1][2]->a[2][3]　　[2][2]->a[2][2]

a[1][3]->a[3][3]　　[2][3]->a[3][2]

....

a[ ][ ]=a[ ][ ]

```
#include<iostream>
using namespace std;
int main()
{
    int n,m,sum;
    int a[101][101];
    cin>>n>>m;
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)
            cin>>a[i][j];
```

```
return 0;
}
```

2 [2683] 图像相似度

给出两幅相同大小的黑白图像（用 0-1 矩阵）表示，求它们的相似度。说明：若两幅图像在相同位置上的像素点颜色相同，则称它们在该位置具有相同的像素点。两幅图像的相似度定义为相同像素点数占总像素点数的百分比。

输入
第一行包含两个整数 m 和 n，表示图像的行数和列数，中间用单个空格隔开。1≤m≤100，1≤n≤100。
之后 m 行，每行 n 个整数 0 或 1，表示第一幅黑白图像上各像素点的颜色。相邻两个数之间用单个空格隔开。
之后 m 行，每行 n 个整数 0 或 1，表示第二幅黑白图像上各像素点的颜色。相邻两个数之间用单个空格隔开。

输出
一个实数，表示相似度（以百分比的形式给出），精确到小数点后两位。

样例输入
3 3
1 0 1
0 0 1
1 1 0
1 1 0
0 0 1
0 0 1
样例输出
44.44

思路：定义两个二维数组，分别表示两幅图像的像素值。依次比较对应像素点的值即可。

```
#include<iostream>
using namespace std;
int main(){
    int a[100][100],b[100][100];
    int m,n,i,j;
    double sum,count=0;
    scanf("%d%d",&m,&n);
    sum=m*n;//计算总点数
    for(i=0;i<m;i++)
        for(j=0;j<n;j++)//输入图像 a
            scanf("%d",&a[i][j]);
```

```
    //输入图像 b,并比较
```

```
    printf("%.2lf",count/sum*100);
    return 0;
}
```

3 [2688] 图像模糊处理

给定 m 行 n 列的图像各像素点的灰度值，要求用如下方法对其进行模糊化处理：

1.四周最外侧的像素点灰度值不变；

2.中间各像素点新灰度值为**该像素点及其上下左右相邻四个像素点原灰度值的平均(舍入到最接近的整数)。**

输入

第一行包含两个整数 n 和 m，表示图像包含像素点的行数和列数。1≤n≤100，1≤m≤100。接下来 n 行，每行 m 个整数，表示图像的每个像素点灰度。相邻两个整数之间用单个空格隔开，每个元素均在 0~255 之间。

输出

m 行，每行 n 个整数，为模糊处理后的图像。相邻两个整数之间用单个空格隔开。

样例输入	样例输出
4 5 100 0 100 0 50 50 100 200 0 0 50 50 100 100 200 100 100 50 50 100	100 0 100 0 50 50 80 100 60 0 50 80 100 90 200 100 100 50 50 100

思路：定义两个二维数组，一个原数组 a，一个模糊后的数组 b，将模糊后的灰度值赋值给 b 数组即可。

1. 如果当前像素的灰度值为 a[i][j]，则其相邻四个方向的灰度值为：

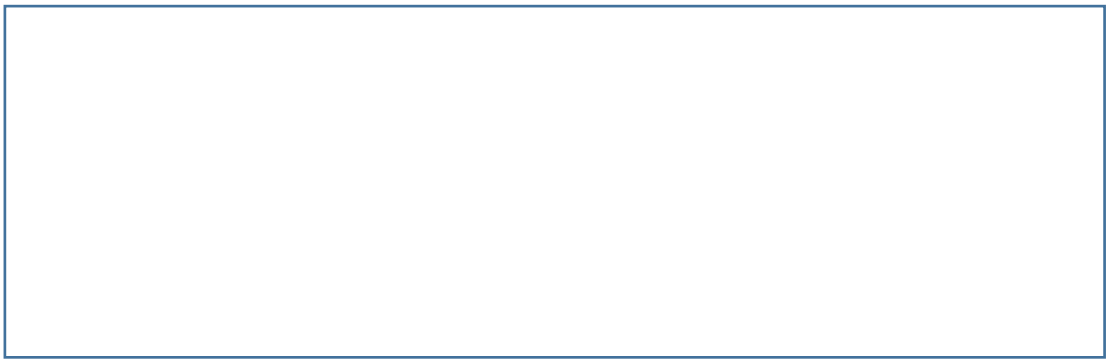
上面 _____下面 _____左边 _____右边 _____

当前点模糊处理后的像素值为_____

2. 四舍五入怎么表示？ int((double)b[i][j]+0.5)

```
#include<iostream>
using namespace std;
```

```
int main(){
    int a[105][105],i,j,b[105][105];//行数 m 和列数 n
    int m,n,sum=0;
    cin>>m>>n;
    for(i=1;i<=m;i++)
        for(j=1;j<=n;j++){
            cin>>a[i][j];
            _____________________;//把数组 a 赋值给数组 b
        }
   //进行模糊处理
```



```
for(i=1;i<=m;i++){
        for(j=1;j<=n;j++)cout<<b[i][j]<<" ";
        cout<<endl;
    }
    return 0;
}
```

4 [2513] 过滤图像

题目描述

图像过滤是把图像中不重要的像素都染成背景色，使得重要部分被凸显出来。现给定一幅黑白图像，要求你将**灰度值位于某指定区间内的所有像素颜色都用一种指定的颜色替换**。

输入

输入在第一行给出一幅图像的分辨率，即两个正整数 M 和 N（0 < M, N <= 500），另外是待过滤的灰度值区间端点 A 和 B（0 <= A < B <= 255）、以及指定的替换灰度值。随后 M 行，每行给出 N 个像素点的灰度值，其间以空格分隔。所有灰度值都在[0, 255]区间内。

输出

输出按要求过滤后的图像。即输出 M 行，每行 N 个像素灰度值，每个灰度值占 3 位（例如黑色要显示为 000），其间以一个空格分隔。行首尾不得有多余空格。

样例输入	样例输出
3 5 100 150 0 3 189 254 101 119 150 233 151 99 100 88 123 149 0 255	003 189 254 000 000 000 233 151 099 000 088 000 000 000 255

思路：_____

_____

```
#include<bits/stdc++.h>
using namespace std;
int a[505][505];
int main()
{
    int m,n;
    int l,r,mid;
    cin>>m>>n;//M 行，每行 N 个像素灰度值
    cin>>l>>r>>mid;//过滤的灰度值区间端点、替换灰度值
    for(int i=1;i<=m;i++)
        for(int j=1;j<=n;j++)
            cin>>a[i][j];
    for(int i=1;i<=m;i++)
    {
        for(int j=1;j<=n;j++)
        {
            if(a[i][j]>=l&&a[i][j]<=r)
            {
                printf("%03d ",mid);
            }
            else
               printf("%03d ",a[i][j]);
        }
        cout<<endl;
    }
}
```

5 *[2682]鞍点问题

给定一个 5*5 的矩阵，每行只有一个最大值，每列只有一个最小值，寻找这个矩阵的鞍点。鞍点指的是矩阵中的一个元素，它是所在行的最大值，并且是所在列的最小值。

```
#include<iostream>
using namespace std;
int main(){
    int a[6][6];
    int maxn,maxcol,maxrow,flag;
    for(int i=1;i<=5;i++)
        for(int j=1;j<=5;j++)
            scanf("%d",&a[i][j]);
    for(int i=1;i<=5;i++){
```

```
            _____;//初始化
            for(int  j=1;j<=5;j++){
                
            }//找出当前行的最大值
            for(int  k=1;k<=5;k++){
                
             }//判断是否存在鞍点
            if(flag==1){//存在
                printf("%d %d %d",maxrow,maxcol,maxn);
                break;
            }
        }
        if(flag==0)//不存在
            printf("not found");
        return 0;
}
```

课堂作业列表

[2046] 杨辉三角　　[1452] 二维数组转置
[2243] 判断上三角矩阵　　[2679] 矩阵交换行
[2687] 图像旋转　　[2683] 图像相似度
[2688] 图像模糊处理　　[2682]鞍点问题
[2680] 同行列对角线的格

课前练习

1 阅读程序	2 #include <iostream>

#include <iostream>
using namespace std;
int main()
{
int i,j,n;
int b[11];
	using namespace std;
	int main()
	{
	int a[9][9];
	int i, j, sum;
	for (i = 1; i <= 8; i++)

```
    n=2016;
    j=0;
    while(n>0)
    {
        j=j+1;
        b[j]=n%3;
        n=n/3;
    }
    for(i=j; i>=1; i--)
        cout<<b[i];
    cout<<endl;
    return 0;
}
```

输出：2202200

```
        for (j = 1; j <= 8; j++)
            cin >> a[i][j];
    sum = 0;
    for (i = 1; i <= 8; i++)
        for (j = 1; j <= 8; j++)
            if (i == j || (i + j == 9))
                sum = sum + a[i][j];
    cout << sum << endl;
    return 0;
}
```

输入：

```
2 3 4 5 6 1 0 9
4 5 6 2 1 4 9 4
9 8 7 6 4 5 2 3
5 6 7 8 2 1 2 3
8 9 0 0 3 4 5 2
9 8 5 6 7 8 9 0
2 3 2 2 2 1 2 3
6 6 6 6 6 6 6 6
```

输出：80

习题

1. 有以下程序：

```
#include<bits/stdc++.h>
using namespace std;
main( )
{  int  x[4][4]={{1,2,3},{3,4,5},{4,5,6},{4,5,6,7}},i,j,t=0;
    for(i=0;i<4;i++ )
    for( j=i;j<4;j++ ) t=t+x[i][j];
    printf("%d\n",t);
}
```

程序运行后的输出结果是(　　A　　)

A)28　　B)45　　C)18　　D)55

2.以下程序运行后的输出结果是　____3 3 2 2_____

```
#include<bits/stdc++.h>
using namespace std;
main( )
```

```
{    int  i,a[10]={  0,0,1,3,1,2,2,1,0,3 },b[4]={0};
    for( i=0;i<10;i++ ) b[a[i]]++;
    for( i=0;i<4;i++ )  printf("%d ",b[i]);
    return 0;
}
```

3. 该程序的运行结果是_____min=-5.770000,m=2,n=1_____

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   double array[4][3]={
       {3.4,-5.6,56.7},
       {56.8,999.,-.0123},
       {0.45,-5.77,123.5},
       {43.4,0,111.2}
    };
    int i,j;
    double min;
    int m,n;
    min = array[0][0];
    m=0;n=0;
    for(i=0;i<3;i++)
    for(j=0;j<4;j++)
      if(min > array[i][j])
      {
        min = array[i][j];
        m=i;n=j;
      }
    printf("min=%lf,m=%d,n=%d\n",min,m,n);
    return 0;
}
```

4. 阅读程序

```
#include <iostream>
using namespace std;
int main()
{
    int n, c, i, j, t, temp;
    int a[51];
    cin >> n;
    for (i = 1; i <= n; i++)
        cin >> a[i];
    for (i = 1; i <= n - 1; i++)
```

```
    {
        c = a[i];
        t = i;
        for (j = i + 1; j <= n; j++)
        {
            if (c < a[j])
            {
                t = j;
                c = a[j];
            }

        }
        if (t != i)
        {
            temp = a[i];
            a[i] = a[t];
            a[t] = temp;
        }
    }
    for (i = 1; i <= n; i++)
        cout << a[i] << " ";
    return 0;
}
```

输入：

18

90 12 33 44 77 29 8 3 4 6 2 1 21 24 23 54 53 25

输出： 90 77 54 53 44 33 29 25 24 23 21 12 8 6 4 3 2 1

第八讲 字符数组

重点
1. 掌握字符数组的含义。
2. 掌握字符串的输入、遍历、输出。
3. 了解字符串。
4. 了解字符串函数。

数组中的每一个元素都存放一个字符数据的数组叫做字符数组。C 语言没有字符串数据类型，对字符串数据的处理都是通过字符数组或指向字符的指针变量来处理。C++可以用 string 来处理。

字符数组的定义形式与前面介绍的数值数组相同。例如，

```
char ch[10];              //定义 ch 为字符数组，包含 10 个元素
char xh[10][20];          //定义 xh 为 10*20 的二维字符数组
char ch[10] = {'c',' ','p','r','o','g','r','a','m'};
```

字符串是由双引号括起来的多个字符，如"This is a book"。C++语言中，字符串的处理可使用字符数组和指向字符的指针变量。实际上，字符串就是一种字符型数组，并且这个数组的最后一个元素是一个字符串结束标志'\0'，也就是说字符串是一种以'\0'结尾的字符数组。1）字符数组可以用字符串来初始化，例如，

```
char c[] = {"Good!"};
```

也可不要花括号，即按下面形式初始化。

```
char c[] = "Good!";
```

8.1 字符数组的输入输出

8.1.1 cin

将整个字符串一次输入或输出。例如有以下程序段：

```
char str [20] ;
cin>>str;        //用字符数组名输入字符串
```

在运行时输入一个字符串，如

```
China↙           在 5 个字符的后面自动加了一个结束符' \0' 。
```

(1)cin.get(字符变量名)可以用来接收字符

#include <iostream>
using namespace std;
main ()
{
  char ch;
	#include <iostream>
	using namespace std;
	main ()
	{
	  char a[20];
		#include <iostream>
		using namespace std;
		main ()
		{
		char m[20];

ch=cin.get();
//或者 cin.get(ch);
cout<<ch<<endl;
}
输入：jljkljkl
输出：j
	cin.get(a,20);
	cout<<a<<endl;
	}
	输入：abc abcdefghijkkl ak
	输出：abc abcde
	（接收 9 个字符+1 个'\0'）
		cin.getline(m,5);
		cout<<m<<endl;
		
		}
		输入：abc aaabbbcccdddee
		输出：abc aaabb
		（跟 get 功能一样）

(2) cin.get(字符数组名,接收字符数目)

用来接收一行字符串,可以接收空格，如上述代码所示。

(3) cin.get(无参数)

没有参数主要是用于舍弃输入流中的不需要的字符,或者舍弃回车,弥补 cin.get(字符数组名,接收字符数目)的不足。

(4) cin.getline()

接受一个字符串，可以接收空格并输出

8.1.2 scanf()

格式：scanf(“%s”,字符串名称);

说明：

①这里的字符串名称之前不加&这个取地址符。例如：scanf(“%s”,&s1)是错误的。

②系统会自动在输入的字符串常量后添加‘\0’标志，因此输入时，仅输入字符串的内容即可。

③输入多个字符串时，以空格分隔。

例如：cin>>s1>>s2>>s3，scanf(“%s%s%s”,s1,s2,s3)；从键盘分别输入 Let us go，则三个字符串分别获取了三个单词。反过来可以想到，如果仅有一个输入字符串名称的情况下，字符串变量仅获取空格前的内容。

例如：scanf(“%s”,s1)；从键盘分别输入 Let us go，则仅有第一个单词被获取，即 s1 变量仅获取第一个单词 Let。

8.1.3 gets()

格式：gets(字符串名称);

说明：使用 gets 只能输入一个字符串。

例如：gets(s1,s2)；是错误的。使用 gets，是从光标开始的地方读到换行符也就是说读入的是一整行，而使用 scanf 是从光标开始的地方到空格，如果这一行没有空格，才读到行尾。

例如：scanf(“%s”,,s1)；gets(s2)；对于相同的输入 Hello World!。s1 获取的结果仅仅是 Hello，而 s2 获取的结果则是 Hello World!

8.1.4 getchar()

getchar 函数的作用是从键盘输入一个字符。这个函数没有参数；返回值为读入字符的

ASCII 编码值。该函数的原型如下：

```
char c1 = getchar( );   //读入一个字符，假设输入小写字母
c1 = c1 - 32;        //转换成大写字母
char c2 = putchar(c1);//用 putchar 输出字符,并把返回的字符赋值给 c2
```

getchar 函数只能读入一个字符，但是如果把 getchar 函数放到循环里，也能实现读入一串字符。例如“要读入一串字符，直到按下回车键表示结束”，可以使用下面的结构：

```
char ch;
while( ( ch=getchar( ) ) != '\n' )
{ …    //处理该字符 ch  }
```

注解：上述循环的执行过程是：先从键盘上读入一个字符，把该字符赋值给 ch，然后判断 ch 是否为换行符'\n'，如果是，循环就结束了；如果不是，则执行循环体。因此该结构可以读入多个字符，一直到回车为止。

8.1.5 字符串的输出

字符数组的输入输出可以有两种方法：

(1) 逐个字符输出。

```
for(i=0;i<len;i++)
    cout<<str[i];
```

(2) 将整个字符串一次输入或输出。例如有以下程序段：

```
char str [20] ;
cin>>str;                  //用字符数组名输入字符串
cout<<str;                 //用字符数组名输出字符串
```

(3) 字符串输出函数 puts( )

puts 使用格式：puts(st)

其中，st 可以是已定义的字符数组名，也可以是指向字符变量的指针变量。功能：把字符数组中的字符串或指针变量所指字符串输出到显示器，即在屏幕上显示该字符串。

例如，

```
char s[6]="China";          // 此处也可写成 char *s="China";
puts(s);                    // puts 不需要格式控制符，输出完后且自动换行
```

等价于：

```
printf("%s\n",s);           // printf 需要格式控制符%s
```

注意：如果 st 是字符数组名，则该函数输出字符数组的第 1 个字符到遇到第一个结束标志之间的所有字符。如果 st 是指向字符串的指针，则该函数输出从 st 所指向的字符到字符串结束标志'\0'之间的所有字符。

例如，若有定义“char *s="Chi\0na"”;，则

语句 puts(s);的输出结果为：Chi

语句 puts(s+1);的输出结果为：hi

语句 puts(s+4);的输出结果为：na

8.2 字符串的访问

输入字符串时，我们没法确定字符串的长度，所以在访问字符串时要明确字符串的长

度或结束状态。通常访问字符串的方法有两种：

（1)通过结束符 ‘\0’来表示结束

```
char str[255];
 cin>>str;
 for(i=0;str[i]!='\0';i++){
    //处理
}
```

(2) 通过获取字符串长度进行访问

```
char str[255];
cin>>str;
len=strlen(str);
for(i=0;i<len;i++){
    //处理
}
```

1 [2556] 态度 100 分

网络上一个经典的帖子：如果将字母 A 到 Z 分别编上 1 到 26 的分数(A=1, B=2, ..., Z=26)，你的知识(KNOWLEDGE)得到 96 分(11 + 14 + 15 + 23 + 12 + 5 + 4 + 7 + 5 = 96)，你的努力(HARDWORK)也只得到 98 分(8 + 1 + 18 + 4 + 23 + 15 + 18 + 11 = 98)，你的态度(ATTITUDE)才是左右你生命的全部(1 + 20 + 20 + 9 + 20 + 21 + 4 + 5 = 100)。
现要求你编写一个程序，实现：从键盘输入任意一单词（假定该单词中只包含大写字母，不需判断），回车，计算并输出该单词的得分。

方法一：

```
#include<bits/stdc++.h>
//#include<cstring>
using namespace std;
int main()
{
        char str[80];//定义一个字符串，用字符数组的方式
        int sum=0;//初始化
        cin>>str;//输入这个字符串的常用方法
    for(int i=0;str[i]!='\0';i++)//遍历字符串
       //len=strlen(str); for(int i=0;i<len;i++)
                sum=sum+str[i]-'A'+1; //或者  sum=sum+str[i]-64
        cout<<sum<<endl;
        return 0;
}
```

方法二：

```
#include <iostream>
using namespace std;
int main()
```

```
{
        int sum = 0;
        char c;
        while((c=getchar())!='\n' && c!=EOF){
                sum+=c-64;
        }
        cout<<sum<<endl;
        return 0;
}
```

2 [2711] 统计数字字符个数

题目描述
输入一行字符，统计出其中数字字符的个数。
输入：一行字符串，总长度不超过 255。
输出：输出为 1 行，输出字符串里面数字字符的个数。

样例输入
Peking University is set up at 1898.
样例输出

```
#include<iostream>
using namespace std;
int main(){
        char str[80];
        int k=0;
        gets(str);
        for(int i=0;str[i]!='\0';i++)
                if(str[i]>='0'&&str[i]<='9')
                        k++;
        cout<<k<<endl;
        return 0;
}
```

8.3 字符数组的初始化

字符数组的初始化与前面介绍的数值数组一样，可通过定义时初始化或通过赋值语句初始化。
1）逐个元素初始化，当初始化数据少于数组长度，多余元素自动为“空”（二进制 0）。当初始化数据多于元素个数时，将出错。例如，

```
char ch[10] = {'c',' ','p','r','o','g','r','a','m'};
```

给前 9 个元素赋初值，最后一个系统自动加上'\0'。数组在内存中的存放形式如图 5-1 所示。

ch[0]	ch[1]	ch[2]	ch[3]	ch[4]	ch[5]	ch[6]	ch[7]	ch[8]	ch[9]
c		p	r	o	g	r	a	m	\0

图 6-1　字符数组在内存中的存放形式

```
char d[2][10]={ { 'I',' ','a','m',' ','a',' ','b','o', 'y'},{'G','o', 'o','d',' ','b','o','y'}};
```

二维数组 d 初始化后，在内存中的存放形式如图 5-2 所示。

d[0][0]	d[0][1]	d[0][2]	d[0][3]	d[0][4]	d[0][5]	d[0][6]	d[0][7]	d[0][8]	d[0][9]
I		a	m		a		b	o	Y
G	o	o	d		b	o	y	\0	\0
d[1][0]	d[1][1]	d[1][2]	d[1][3]	d[1][4]	d[1][5]	d[1][6]	d[1][7]	d[1][8]	d[1][9]

图 6-2　二维数组 d 在内存中的存放形式

2）指定初值时，若未指定数组长度，则长度等于初值个数。

```
char c[ ] = {'I',' ','a','m',' ','h','a','p','p','y'};
```

等价于：

```
char c[10] = {'I',' ','a','m',' ','h','a','p','p','y'};
```

3）使用赋值语句逐个元素赋值，例如，

```
char c[10];
c[0]='I'; c[1]=' '; c[2]='a'; c[3]='m'; c[4]=' ';
c[5]='h'; c[6]='a'; c[7]='p'; c[8]='p'; c[9]='y';
```

8.4　字符串初始化

1）字符数组可以用字符串来初始化，例如，

```
char c[] = {"Good!"};
```

也可不要花括号，即按下面形式初始化。

```
char c[] = "Good!";
```

字符数组在内存的存放形式如图 4-3 所示。

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]
G	o	o	d	!	\0

图 6-3　字符串初始化的数组在内存中的存放形式

2）字符串在存储时，系统自动在其后加上结束标志（占 1 字节，其值为二进制 0）。但字符数组并不要求其最后一个元素是'\0'，例如，要注意下面数组使用的区别。

```
char c1[5]={'G','o','o','d','!'};
char c2[]={"Good!"};
```

注意：字符串常量只能在定义字符数组时赋初值给字符数组，不能将 1 个字符串常量直接赋值给字符数组。例如，下面的使用方法是错误的。

```
char st[5];
st={"Good!"};
st="Good!";
```

这是因为 st 是数组名，不能直接被赋值，要将一个字符串常量“赋值”给一个字符数组，可以使用 5.6.3 节介绍的字符串处理函数来实现。

8.5 案例讲解

3[2722] 字符串逆序

输入一串长度不大于 200 的包含字符‘!’的字符串，取出碰到第一个“！”的字符串，将其逆序输出。如测试数据所述
样例输入 abc!aa
样例输出 cba

```
#include <iostream>
using namespace std;
int main(){
    int _1__________;
    char a[200];
    _2__________
    while(_3__________){
        i++;
    }
    i--;
    for(_4__________)
    printf("%c",a[i]);
    return 0;
}
```

4 [2691] 基因相关性

为了获知基因序列在功能和结构上的相似性，经常需要将几条不同序列的 DNA 进行比对，以判断该比对的 DNA 是否具有相关性。

现比对两条长度相同的 DNA 序列。定义两条 DNA 序列相同位置的碱基为一个碱基对，如果一个碱基对中的两个碱基相同的话，则称为相同碱基对。接着计算相同碱基对占总碱基对数量的比例，如果该比例大于等于给定阈值时则判定该两条 DNA 序列是相关的，否则不相关。

输入：有三行，第一行是用来判定出两条 DNA 序列是否相关的阈值，随后 2 行是两条 DNA 序列(长度不大于 500)。输出：若两条 DNA 序列相关，则输出“yes”，否则输出“no”。

样例输入

0.85

ATCGCCGTAAGTAACGGTTTTAAATAGGCC

ATCGCCGGAAGTAACGGTCTTAAATAGGCC

样例输出

Yes

```
#include <iostream>
#include <cstring>
using namespace std;
int main()
{
    char str1[80],str2[80];
    double k;
    int len,sum=0;
    cin>>k>>str1>>str2;
    len=_1_____________;
    for(int i_2_____________)
        if(__3___________)sum++;
    if(___4_____________)
        cout<<"yes";
    else
        cout<<"no";
    return 0;
}
```

5 *[3277] 扫雷游戏

扫雷游戏是一款十分经典的单机小游戏。在 n 行 m 列的雷区中有一些格子含有地雷（称之为地雷格），其他格子不含地雷（称之为非地雷格）。玩家翻开一个非地雷格时，该格将会出现一个数字——提示周围格子中有多少个是地雷格。游戏的目标是在不翻出任何地雷格的条件下，找出所有的非地雷格。

现在给出 n 行 m 列的雷区中的地雷分布，要求计算出每个非地雷格周围的地雷格数。

注：一个格子的周围格子包括其上、下、左、右、左上、右上、左下、右下八个方向上与之直接相邻的格子。

对于 100%的数据， 1≤n≤100， 1≤m≤100。

输入文件第一行是用一个空格隔开的两个整数 n 和 m，分别表示雷区的行数和列数。接下来 n 行，每行 m 个字符，描述了雷区中的地雷分布情况。字符’*’表示相应格子是地雷格，字符’?’表示相应格子是非地雷格。相邻字符之间无分隔符。

输出文件包含 n 行，每行 m 个字符，描述整个雷区。用’*’表示地雷格，用周围的地雷个数表示非地雷格。相邻字符之间无分隔符。

样例输入

3 3
*??
???
?*?
样例输出
*10
221
1*1
方法一:
```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int m,n,k=0,i,j;
    char a[105][105];
    cin>>n>>m;
    for(i=0;i<n;i++)
        for(j=0;j<m;j++)
            cin>>_1____________;
    for(i=0;i<n;i++){
        for(j=0;j<m;j++){//计算每一行
            ___2___________________;
            if(a[i][j]=='*')cout<<"*";//判断每一个位置
            else{
                if(a[i-1][j-1]=='*')k++;
                if(a[i-1][j]=='*')k++;
                if(_3____________=='*')k++;
                if(_4____________=='*')k++;
                if(_5____________=='*')k++;
                if(_6____________=='*')k++;
                if(_7____________=='*')k++;
                if(_8____________=='*')k++;
                cout<<k;
            }

        }
        __9___________
    }
    return 0;
}
```
方法二:
```
#include<iostream>
using namespace std;
char a[101][101];
int next1[8][2]={_____________________________________};
```

```
int main(){
    int m,n,i,j,k,num;
    int dx,dy;
    cin>>n>>m;
    for(i=1;i<=n;i++)
        for(j=1;j<=m;j++)
            cin>>a[i][j];
    for(i=1;i<=n;i++){
        for(j=1;j<=m;j++){
            if(a[i][j]=='*')cout<<'*';
            else{
                num=0;
                for(k=0;k_____________________){
                    dx=i+_____________________;
                    dy=j+_____________________;
                    if(a[dx][dy]=='*')
                        num++;
                }
                cout<<num;
            }
        }
        cout<<endl;
    }
    return 0;
}
```

课堂作业列表

[2556] 态度 100 分　　[2711] 统计数字字符个数
[2722] 字符串逆序　　[2691] 基因相关性
[3277] 扫雷游戏　　2695 配对碱基链
2696 密码翻译

课前练习

1 阅读程序
#include <iostream>
using namespace std;
int main()
{
    int i,n,r,s,x;
    cin>>n;
    for(i=1; i<=n; i++)
	#include <iostream>
	using namespace std;
	int main()
	{
	    int i, j, max;
	    int a[9], s[9];
	    max = -32765;
	    for (i = 0; i <= 8; i++)s[i] = 0;

```
    {
        cin>>x;
        s=0;
        while(x!=0)
        {
            r=x%2;
            if(r==1)
                s=s+1;
            x=x/2;
        }
        cout<<s<<endl;
    }
    return 0;
}
```

输入：
4 2 100 1000 66
输出：_____
1
3
6
2

```
    for (i = 1; i <= 8; i++)
    {
        cin >> a[i];
        s[i] = s[i - 1] + a[i];
    }
    for (i = 0; i <= 7; i++)
        for (j = i + 1; j <= 8; j++)
        {
            if ((s[j] - s[i]) > max)
                max = s[j] - s[i];
        }
    cout << max << " ";
    for (i = 1; i <= 7; i++)
        cout << s[i] << " ";
    return 0;
}
```

输入：46 36 46 11 28 28 21 18
输出：__234 46 82 128 139 167 195 216_

习题

1.合法的数组定义是(　　)。

A. char a[ ]= "string " ;　　B. int a[5] ={0,1,2,3,4,5};

B. char a= "string " ;　　D. char a[ ]={0,1,2,3,4,5}

2. 若要求从键盘读入含有空格字符的字符串，应使用函数(　　)。

A）getc()　B) gets()　C) getchar()　D) scanf()

3. 下面哪一项是不正确的字符串赋值或赋初值的方式(　　)。

A. char *str; str="string";

B. char str[7]={'s','t','r','i','n','g'};

C. char str1[10];str1="string";

D. char str1[]="string",str2[]="12345678";

4. 设有定义：char s[12] = "string" ;　则 printf("%d\n",strlen(s));　的输出是(　　)。

A. 6　　B. 7　　C. 11　　D. 12

5. 以下不正确的初始化形式是(　　)。

A)char c[]={"boy"};　　B)char c[]="boy";

C)char c="boy";　　D)char c[6]={'b', 'o', 'y', '\0'};

6. 写出下面这个程序的输出结果:

```
int main()
```

```
{
  char str[]="ABCDEFGHIJKL";
  printf("%s\n",str);       屏幕上显示_________
  printf("%s\n",&str[4]);   屏幕上显示_________
  str[2]=str[5];
  printf("%s\n",str);       屏幕上显示__________
  str[9]='\0';
  printf("%s\n",str);       屏幕上显示__________
}
```

7. 输出:___abcdefghi__________

```
int main( )
{
   char s[80]="abcdefghi";
      int i ;
   for(i=0; i<80; i++)
         if(s[i]=='\0') break;
      s[i]='\0';  i=0;
      while(s[i])  putchar(s[i++]);
      putchar('\n');
}
```

8. 对给定的一个字符串，把其中从 a-y，A-Y 的字母用其后继字母替代，把 z 和 Z 用 a 和 A 替代，其他非字母字符不变，则可得到一个简单的加密字符串。

```
#include<cstdio>
#include<cstring>
using namespace std;
int main(){
    char a[90];
    int i,n;
    gets(a);
    ___1_________
    for(i=0;i<n;i++){
        if(a[i]=='Z'||a[i]=='z') ____2_________
        else if(a[i]>='a'&&a[i]<='z'||a[i]>='A'&&a[i]<='Z') ___3____
        else a[i]=a[i];
    }
    printf("%s",a);
return 0;}
```

第九讲 数组与指针

重点

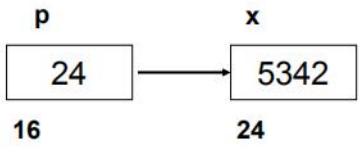
1. 理解地址的含义。
2. 理解指针的定义。
3. 了解数组名的含义。

什么是地址？地址就是变量在内存中的存储位置。数组在内存占用一片连续的存储空间，这个连续空间的首地址为数组的指针，即数组的指针是指数组的首地址。数组元素的地址称为数组元素的指针。因此，同样可以用指针变量来指向数组或数组元素。

9.1 指针

指针是 C++ 语言的一个重要概念，也是 C++ 语言的重要特色。C++ 语言的高度灵活性及极强的表达能力，在很大程度上表现在巧妙而灵活的运用指针。

例 1：几位同学去玩某密室逃脱游戏，密室门上有一把四位数的数字密码锁，只有在室中找到开锁密码才能走出密室。密室中整齐地摆放着规格大小完全相同的 26 个寄在箱，每个寄存箱上按顺序都有一个英文字母和一个编号，字母从 A 到 Z，编号从 1 到 26。同学们在密室中认真搜寻，在角落里找到一把钥匙，钥匙上刻着字母 P，用这把匙打开 P 寄存箱(编号为 16)，里面是一把刻着数字 24 的钥匙，用这把钥匙再打开号为 24 的 X 寄存箱，里面有一张字条，上面写着“5342”。用这四个数字去开密码镜果然打开了,成功逃脱密室。



名字	p	x
地址	16	24
内容	24	5342

上述过程可以用图 1 来表示。

图 1 密码存放示意图

分析： 每个寄存箱都有一个名字和一个编号(即地址)，可以通过名字找到寄存箱(如第一把钥匙上的 P)，也可以通过编号(地址)访问寄存箱(如刻着数字 24 的匙)。在此游戏中，P 寄存箱比较特殊,不是直接保存密码值,而是存放了另一个寄存箱的编号(地址)，同学们是根据线索顺藤摸瓜,通过 P 寄存箱间接找到 X 寄存箱中的密码的。可以使用指针把上述故事写成下面的 C 程序：

获取密码的两种方法 ：

```
#include<iostream>
using namespace std;
int main( )
{
    int x = 5342;      /* 变量 x 用于存放密码值 5342 */
    int *p = NULL;    /* 定义整型指针变量 p，NULL 值为 0，代表空指针 */
    p = &x;          /*将变量 x 的地址存储在 p 中 */
     /* 通过变量名 x 输出密码值*/
    printf("通过变量名 %d\n ", x);
         /* 通过变量 x 的地址输出密码值 */
```

```
    printf("通过地址 %d\n",p, *p);
    return 0;
}
```

程序中定义了变量 x 来存放密码，再定义一个特殊的指针变量 p，用于存放变量 x 的地址。这样既可以通过变量名 x 直接得到密码值,也可以在不知道变量名的情况下通过指针变量 p 所存放的 x 的地址间接找到密码值。

指针变量的定义格式为：

数据类型 *指针变量;

可以通过赋值语句给指针变量赋值，例如“p = &a；”表示把变量 a 的内存地址赋值给 p。

指针变量初始化：

int *p = NULL;

对于“int a = 3;”，系统会在内存的某个区域开辟连续 4 个字节的单元存储。对 a 的操作就是对该内存区域进行操作，至于是具体的哪 4 个单元，我们并不关心。

内存单元的位置（编号）叫作“地址”，可以通过取地址操作符“&”获得一个变量 a 的起始地址（首个存储单元的地址）：&a。

指针也是一个变量。和普通变量不同的是，指针变量里存储的数据是一个内存地址，就好像一个指示器，指引着你去该内存地址开始的一块内存区域存取数据。

9.2 指向一维数组的指针

1．用指针引用数组元素

假设定义一个一维数组，系统将给该数组在内存中分配的一个存储空间，C++语言规定其数组名就是数组在内存中的首地址。若再定义一个指针变量，并将数组的首地址传给指针变量，则该指针就指向了这个一维数组。可以说数组名是数组的首地址，也就是数组的指针。而定义的指针变量就是指向该数组的指针变量。对一维数组的引用，既可以下标法，也可使用指针的表示方法。

int a[10] , *p; //定义数组与指针变量* /

做赋值操作：

p=a;　或　p=&a[0];

则 p 就得到了数组的首地址。其中，a 是数组的首地址，&a[0]是数组元素 a[0]的地址，由于 a[0]的地址就是数组的首地址，所以，两条赋值操作效果完全相同。指针变量 p 就是指向数组 a 的指针变量，如图 1 所示。

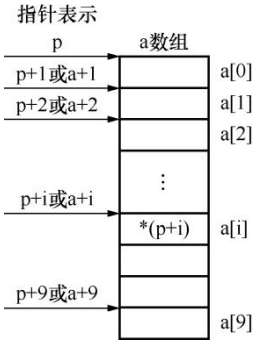


图 1　一维数组的指针法访问

当使指针 p 指向数组 a 后，可以用指针 p 访问数组的各个元素。如果指针 p 指向数组 a（指向数组的第一个元素 a[0]），则

p+1 指向下一个元素 a[1]

p+i 指向元素 a[i]

注意：p＋1 不是将 p 值简单加 1。如果数组元素是整型，p＋1 表示 p 的地址加 2；如果数组元素是实型，p＋1 表示 p 的地址加 4；如果数组元素是字符型，p＋1 表示 p 的地址加 1。使用指针法引用一维数组的第 i 个元素的方法有如下 3 种。

1）*(p+i)访问元素 a[i]。

2）*(a+i)访问元素 a[i]。

3）指向数组的指针变量也可以带下标，即 p[i]与*(p+i)等价，表示元素 a[i]。

例 1 使用指针法实现：输入 N 个数据存入数组中，输出其中的最大元素。

方法一：通过数组名计算数组元素的地址，引用数组元素。

```
#include<iostream>
using namespace std;
const int N=10;
int main()
{    int i,imax,max,a[N];                    //imax 代表最大值元素所在的位置
     for(i=0;i<N;i++)
       scanf("%d",a+i);
    max = a[0];                          //假设第 0 个元素就是最大元素
    imax= 0;
    for(i=1;i<N;i++)
          if (*(a+i) > max)
             {
                max = *(a+i);
               imax = i;
          }
    printf(" The Max Number a[%d]=%d\n",imax,max);
 }
```

方法二：使用指针变量作循环控制变量。

```
#include<iostream>
using namespace std;
#define  N 10
int main()
{
  int i,imax,max,a[N],*p;              //imax 代表最大值元素所在的位置
  for(p=a;p<a+N;p++)
      scanf("%d",p);
```

```
    max = a[0];                          //假设第 0 个元素就是最大元素
    imax= 0;
    p=a;              //此语句不能少，因为在输入数据后，指针已指向数组外了
    for(i=0;i<N;i++,p++)  //此语句可以写成 for(i=0;p<a+N;i++,p++)
        if(*p > max)
        { max = *p);
          imax = i;
        }
    printf(" The Max Number a[%d]=%d\n",imax,max);
}
```

这两种方法与下标法的比较如下。

1）方法一与下标法执行效率是相同的，C++编译系统是将 a[i]转换为*(a+i)处理，即先计算元素的地址。

2）方法二比方法一和下标法执行效率高，用指针变量直接指向元素，不必每次都重新计算地址，使用指针运算 p++指向下一个元素，这种有规律地改变地址值 p++能大大提高程序执行效率。

3）用下标法比较直观，能直接知道是第几个元素。使用指针法，一定要知道当前指针指向哪个元素，否则可能得到意想不到的结果。

使用指针引用数组元素，应注意以下问题。

1）若指针 p 指向数组 a，虽然 p+i 与 a+i、*(p+i)与*(a+i)意义相同，但仍应注意 p 与 a 的区别（a 代表数组的首地址，是不变的；p 是一个指针变量，可以指向数组中的任何元素），例如，

```
for(p=a; a<(p+10); a++) //错误，因为 a 代表数组的首地址，是不变的，a++不合法
  printf("%d", *a);
```

2）指针变量可以指向数组中的任何元素，注意指针变量的当前值。

例 3　输入 10 个数据存入数组 a 中，然后打印输出数组 a。

```
#include<iostream>
using namespace std;
int main()
{int *p, i, a[10];
   p = a;
   for(i=0;i<10;i++)
       scanf("%d", p++);//输入 2 10 3 20 14 56 70 11 22 78
 for(i=0;i<10; i++,p++)
       printf("%d ", *p);
}
```

输入 ：2 10 3 20 14 56 70 11 22 78

程序运行结果如下。

37 0 0 3 6487624 0 10556288 0 4199400 0

思考与讨论：运行结果显然是不对的，这是什么原因呢？其原因是使用指针访问数组越界。在例 2 中，第 2 次 for 循环开始时，p 已经越过数组的范围，如图所示。C++语言编译器不能发现该问题，避免指针访问越界是程序员自己的责任。此例只需要在第 2 次 for 循环前面加一个语句“p=a;”，即让指针指向数组首地址即可避免。

2．指向数组元素的指针的一些运算

设有定义“int a[10],*p=a;”，则对指向数组的指针变量的一些操作运算如表 4-1 所示。

表 4-1　指向数组元素指针的一些运算

运算操作	说　明
p++（或p += 1）	p指向下一个元素
*p++	相当于*(p++)。因为*和++同优先级，++是右结合运算符
*(p++)	先取*p，再使p加1，即先取得p所指向元素的值，让指针指向下一个元素
*(++p)	先使p加1，再取*p，即先让p+1指向下一个元素，再取指向元素的值
(*p)++	p指向的元素值加1，指针仍指向原来的元素

例 4　指向数组的指针变量的运算示例。

```
#include<iostream>
using namespace std;
Int main()
{ int a[6]={2,4,6,8,10,12},*p;
    p=a+2;
    printf("%d ",*p++);    //输出 p 所指向的元素,即 a[2],让 p 指向下一个元素,即 a[3]
    printf("%d ",*(p++)); //输出 p 所指向元素 a[3],让 p 指向下一个元素 a[4]
    printf("%d ",*++p);    //让 p 指向下一个元素 a[5],输出 p 所指向元素 a[5]的值
    printf("%d ",(*p)++); //输出 p 所指向元素 a[5]的值,让 p 所指向元素的值加 1
    printf("%d ",*p);          //输出 p 所指向元素的值,即 a[5]
}
```

根据程序的注释，不难分析出程序运行的输出结果：
6　8　12　12　13

9.3 案例讲解

1[6881] 数字变化

[问题描述]
输入两个不同的整数，把较小的那个数翻倍并输出。
[输入格式] 一行两个整数（int 范围以内），之间用一个空格隔开。
[输出格式] 一行一个整数，较小数翻倍后的结果。
[输入样例] 2 3

[输出样例] 4

```
#include<iostream>
using namespace std;
int main(){
    int a,b;
    int *p;
    cin >> a >> b;
    if(a < b) _____________________;
    else p = _____________________;
    cout <<_____________________ << endl;
    // 取出 p 指向的内存单元里的整数，乘以 2 输出
    return 0;
}
```

1） 变量 a 和 b 一旦定义，系统就会给它们分配内存空间，而且在程序运行过程中，其内存地址是固定不变的，这种存储方式称为“静态存储”。

2） 指针变量 p 定义后，其地址空间是不确定的，默认是 NULL。当执行到 p = &a 或者 p = &b 时，p 才指向 a 或者 b 的地址，才能确定 p 的值。这种储存方式称为“动态存储”。

3） 指针的动态性，还体现在可以根据需要，通过函数 new() 随时申请。看以下例 2：

2[2404] 指针变量输出

样例输入

```
1 2 3 4 5 6 7 8 9 0
```

样例输出

```
1 2 3 4 5 6 7 8 9 0
```

```
#include<iostream>
using namespace std;
int main()
{
    int *p,a[10],i;
    p=_1_____________;
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(p=a;p_2_____________)
        printf("%d ",__3_____________);
    printf("\n");
    return 0;
}
```

3[6882] 输出最大值

使用指针法实现：输入 N 个数据存入数组中，输出其中的最大元素。

通过数组名计算数组元素的地址，引用数组元素。

```
#include<iostream>
using namespace std;
#define  N 10
int main()
{    int i,imax,max,a[N];                    //imax 代表最大值元素所在的位置
      for(i=0;i<N;i++)
        scanf("%d",a+i);
    max = a[0];                          //假设第 0 个元素就是最大元素
    imax= 0;
    for(i=1;i<N;i++)
          if (*(__________) > max)
             { max =__________;
            imax = i;
          }
    printf(" The Max Number a[%d]=%d\n",imax,max);
 }
```

课堂作业列表

[6881] 数字变化　　[2404] 指针变量输出
[6882] 输出最大值　　2669 开关灯
2680 同行列对角线的格　　7260 乐乐的得分
6858 整理题库

课前练习

```
#include <iostream>
using namespace std;
int main()
{
    long x;
    int i, n;
    cin >> n;
    x = 1;
    for (i = 1; i <= n; i++)
        x = x * i;
    cout << x;
    return 0;
}
```

输入：8
输出：__40320_____

```
int main(  )
{  int a[10]={1,1,2,8,4,2,3,3,8,2};
   int i,j,k,n=0;
   for(i=0; i<10-n; i++)
   {  for(j=i+1; j<10-n; j++)
       if(a[j]==a[i])
       {  for(k=j; k<10-n; k++)
            a[k]=a[k+1];
          n++;
       }
   }
   for(i=0;i<10-n;i++)
              printf("%d ",a[i]);
   printf("\n");
}
```

输出：___1 2 8 4 3_____

习题

1. 若有以下程序段：

```
int a,b,*p,*q;
p=&a;
q=&b;
p=q;
*p=5;
```

则以下叙述中正确的是_____。

A）*p 表示存储单元 a　　B）*p 表示存储单元 b

B）*p 表示变量 a 的地址值　　D）*p 表示变量 b 的地址值

2. 若有初始化“int a=2，*p；”，则要使 p 指向 a 应使用的语句是____。

A）*p=a；　B）*p=&a；　C）p=a；　D）p=&a；

3. 若有定义“int *p,*q, a=2,b;”，则以下正确的赋值语句组是____。

A)p=&a; *q=*p;　　B) *p=a;*q=b;

C) p=&a;q=&b;*p=*q;　　D)p=&a;q=&b;*q=*p;

4. 若有定义“int a[5]={1，2，3，4，5}，*p”，则以下表达式中值为 3 的是_____。

A） p+2　B）*（p+2）　C）*（p+3）　D）*p+3

5. 数组名代表数组的首地址，以下选项中可以对数组名进行的运算符是_____。

A）++　B）+　C）-　D）=

6. 以下程序运行后的输出结果是____。

```
int main(  )
{
    int  x[8]={37,43,56,28,90,13,55,79 },i,j,t;
    i=0;  j=7;
    while(i<j)
    {   t=x[i];x[i]=x[j]; x[j]=t; i++;  j--; }
    for( i=0;i<3;i++ )  printf("%5d",x[i]);
}
```

7.以下程序运行后的输出结果是____。

```
main( )
{   int i,j;
   for(i=0;i<=2;i+=2)
   {   for(j=10;j>=5;j--)
       {   if((j+i)%2)
          { j--; printf("%d ",j); continue; }
          j=j-2;
          printf("%d  ",j);
```

```
8  6 4
8  6 4
```

```
        }
        printf("\n");
    }
}
```

8.指针变量输出

样例输入 1 2 3 4 5 6 7 8 9 0

样例输出 1 2 3 4 5 6 7 8 9 0

```
#include<stdio.h>
int main()
{
    int *p,a[10],i;
    p=__1_____________
    for(i=0;i<10;i++)
        scanf("%d",&a[i]);
    for(_2_____________p++)
        printf("%d ", _3_____);
    printf("\n");
    return 0;
}
```

9.从键盘上输入一行字符，统计其中数字字符的个数。

```
#include <stdio.h>
int main()
{  char str[10];
     int i, __1_____;
       for(i=0;i<10;i++)                    // 输入 10 个字符
      scanf("%c",&str[i]);
     for(i=0;i<10;i++)
      if(__2_____________ )
         digit++;
     printf("The number of digit is %d.\n",digit);
}
```

第十讲　字符串

重点
1. 掌握字符数组的应用。
2. 掌握常见的字符函数。
3. 理解字符统计的用法。

在 c++中，字符串有两种表示方法，一种通过字符数组表示，一种通过 string 类型表示。

```
char str[200];//定义一个字符数组表示字符串
string str;//定义一个字符串类型
```

字符数组的用法前面已经讲过，这里简单介绍 string 的基础用法。

10.1 string 用法

定义字符串还可以用 stirng，为了在程序中使用 string 类型，必须包含头文件<string>。

声明一个字符串变量很简单：string str;

10.1.1 输入输出

在 C++中，在输入输出方面，我们可以像对待普通变量那样对待 string 类型变量，其输入输出仍然可以用输入输出操作符进行处理。

对 string 类型而言，读取时忽略开头所有的空白字符（如空格，换行符，制表符），读取字符直至再次遇到空白字符，读取终止。如果给定输入是"Hello World!"（注意到开头和结尾的空格），则屏幕上将输出"Hello"，而不含任何空格。

```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string s;
    cin>>s;
    cout<<s<<endl;
    return 0;
}
```

输入一行没有空格的字符串时，可用

```
string  str;
cin>>str;
cout<<str;
```

输入一行带空格的字符串时，可用

```
string str;
getline(cin,str);
```

10.1.2 初始化

跟字符数组一样，可以在定义 string 字符串的时候，给字符串赋初始,，举例如下：

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s1; // 默认初始化，s1="","空串，表示里面没有字符
    string s2 = "I Love China"; // 把 I Love China 这个字符串内容拷贝到了 s2 代表的一段内存中，拷贝时不包括末尾的\0;
    string s3("I Love China"); // 与 s2 的效果一样。
    string s4 = s2; // 将 s2 中的内容拷贝到 s4 所代表的的一段内存中。
    int num = 6;
    string s5 = (num,'a');// 将 s5 初始化为连续 num 个字符的'a'，组成的字符串，这种方式不太推荐，因为会在系统内部创建临时对象。
    return 0;
}
```

10.1.3 string 的比较等操作

你可以用 ==、>、<、>=、<=、和!=比较字符串，可以用+或者+=操作符连接两个字符串，并且可以用[]获取特定的字符。阅读以下程序：

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string str;
    cout << "Please input your name:"<<endl;
    cin >> str;
    if( str == "Li" )   // 字符串相等比较
        cout << "you are Li!"<<endl;
    else if( str != "Wang" )  // 字符串不等比较
        cout << "you are not Wang!"<<endl;
    else if( str < "Li")     // 字符串小于比较，>、>=、<=类似
        cout << "your name should be ahead of Li"<<endl;
    else
        cout << "your name should be after of Li"<<endl;
    str += ", Welcome!";  // 字符串+=
```

```
        cout << str<<endl;

        return 0;
    }
```

10.2 获取长度

对于 string 类型，可以通过 length()/size() 获取字符串长度。

```
string str;
for(int i = 0 ; i < str.size(); i ++)
    cout<<str[i];  // 类似数组，通过[]获取特定的字符
```

对于字符数组而言，可以通过，测字符串长度函数 **strlen( )，来获取字符串长度。**

例如，

```
char s[10]="abcde";
printf("%d\n",strlen(s));
```

输出结果为 5。

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string str1;
    char str2[100];
    int i,sum1=0,sum2=0;
    cin>>str1>>str2;//abcda aabbaa
    for(i=0;i<str1.length();i++)
    {
        if(str1[i]=='a')
        {
            sum1++;
        }
    }
    for(i=0;i<strlen(str2);i++)
    {
        if(str2[i]=='a')
        {
            sum2++;
        }
    }
    cout<<sum1<<" "<<sum2;
    return 0;
}
```

10.3 综合练习

1[1455] 顺序输出元音字母

输出字符串中的元音字母
输入:一行字符串
输出:按照输入的顺序输出其中的元音字母
样例输入:abcde
样例输出:ae

分析：
1. **定义字符串** _____
2. **输入字符串** _____
3. **获取字符串长度：**_____
4. **遍历字符串的每个字符：**_____
5. **判断是否是元音字母**_____

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char a[10001];
    cin>>a;//gets(a);
    for(int __1_____________________)//遍历每个字符
    {
        if(a[i]==_____________________)//如果等于元音字符时 a o e u i
        {
            cout<<a[i];
        }
    }
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
    int len;
    int i;
    string str;
    cin>>str;//getline(cin,str);
    len=_____________________// str.length();
        for(int i=0;i<len;i++)
```

```
            if(str[i]=='a'||str[i]=='e'||str[i]=='i'|str[i]=='o'||str[i]=='u')
                cout<<str[i];
    return 0;
}
```

2［2693］输出亲朋字符串

编写程序，求给定字符串 s 的亲朋字符串 s1。

亲朋字符串 s1 定义如下：给定字符串 s 的第一个字符的 ASCII 值加第二个字符的 ASCII 值，得到第一个亲朋字符； 给定字符串 s 的第二个字符的 ASCII 值加第三个字符的 ASCII 值，得到第二个亲朋字符；依此类推，直到给定字符串 s 的倒数第二个字符。亲朋字符串的最后一个字符由给定字符串 s 的最后一个字符 ASCII 值加 s 的第一个字符的 ASCII 值。

输入输入一行，一个长度大于等于 2，小于等于 100 的字符串。**字符串中每个字符的 ASCII 值不大于 63。**

输出一行，为变换后的亲朋字符串。输入保证变换后的字符串只有一行。

样例输入 1234

样例输出 cege

分析：遍历字符串中的每个字符，从左到右两两相加，相加后直接输出即可。

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
    _1_____________//定义字符串
   _2_____________//输入字符串
    int len 3_____________//获取字符串长度
    char res;
    for(_4_____________)//遍历每个字符
    {
        res=_5_____________//将相邻两个字符相加
        cout<<res;//输出
    }
    res=str[len-1]+str[0];//求解最后一个字符
    cout<<res<<endl;
    return 0;
}
```

3［1900］查找最大元素

对于输入的每个字符串，查找其中的最大字母，在该字母后面插入字符串“(max)”。

输入输入数据包括多个测试实例，每个实例由一行长度不超过 100 的字符串组成，字符串仅由大小写字母构成。

输出

对于每个测试实例输出一行字符串，输出的结果是插入字符串“(max)”后的结果，如果存在多个最大的字母，就在每一个最大字母后面都插入"(max)"。

样例输入

abcdefgfedcba

样例输出

abcdefg(max)fedcba

分析：遍历字符串中的每个字符，找出最大字符；重新再遍历每一个字符，如果跟最大字符相等就在后面就上(max).。

```
#include<iostream>
#include<cstring>
using namespace std;
 int main(){
    char a[100],mx;
    int i,len;
    cin>>a;
    _1_____________
    len= _2_____________
    for(i=0;i<len;i++){
       _3_____________________; //求最大值
    }
    for(i=0;i<len;i++){
        cout<<a[i];
        _4_____________________//如果是最大值后面加上(max)
    }
    cout<<endl;
    return 0;
}
```

4［2693］输出亲朋字符串

编写程序，求给定字符串 s 的亲朋字符串 s1。

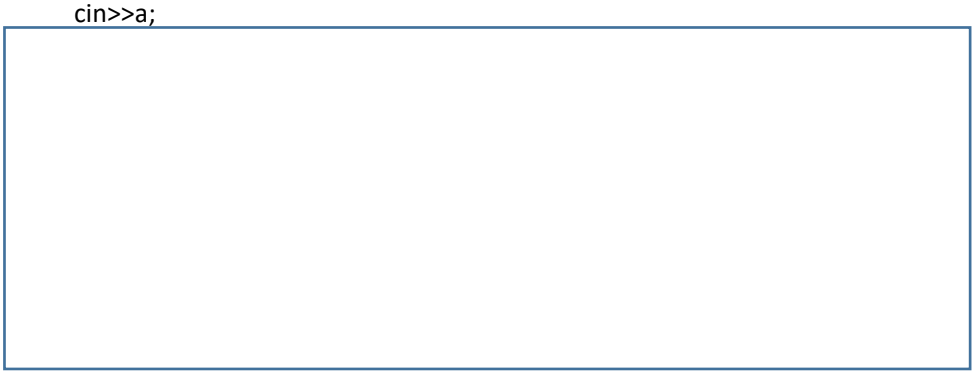
亲朋字符串 s1 定义如下：给定字符串 s 的第一个字符的 ASCII 值加第二个字符的 ASCII 值，得到第一个亲朋字符； 给定字符串 s 的第二个字符的 ASCII 值加第三个字符的 ASCII 值，得到第二个亲朋字符；依此类推，直到给定字符串 s 的倒数第二个字符。亲朋字符串的最后一个字符由给定字符串 s 的最后一个字符 ASCII 值加 s 的第一个字符的 ASCII 值。

输入输入一行，一个长度大于等于 2，小于等于 100 的字符串。字符串中每个字符的 ASCII 值不大于 63。

输出输出一行，为变换后的亲朋字符串。输入保证变换后的字符串只有一行。

样例输入
1234
样例输出
cege

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
    char a[111];
    cin>>a;
```



```
    cout<<res<<endl;
    return 0;
}
```

5 *[2699] 整理药名

医生在书写药品名的时候经常不注意大小写，格式比较混乱。现要求你写一个程序将医生书写混乱的药品名整理成统一规范的格式，即药品名的**第一个字符是字母要大写，其他字母小写**。如将 ASPIRIN、aspirin 整理成 Aspirin。

输入第一行一个数字 n，表示有 n 个药品名要整理，n 不超过 100。
接下来 n 行，每行一个单词，长度不超过 20，表示医生手书的药品名。药品名由字母、数字和-组成。
输出 n 行，每行一个单词，对应输入的药品名的规范写法。

样例输入	样例输出
4 AspiRin cisapride	Aspirin Cisapride 2-penicillin

2-PENICILLIN Cefradine-6	Cefradine-6

思路：按题意要求，判断第一个字符如果是字符，改成大些，其它字符如果是小写，则改成小写字符。

1. 如何判断第一个字母是大写字母？_____
2. 如何把大写字符改成小写字母？_____

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
	char a[111];
	int n;
	cin >> n;
	while (n--)
	{
	}
	return 0;
}
```

课堂作业列表

［1455］顺序输出元音字母
［2690］找第一个只出现一次的字符
［2708］连续出现的字符
［1331］删除字符串中指定字符
1464 加密处理后的字符
1899 C 语言合法标识符
1900 查找最大元素
1901 首字母变大写
3331 你的飞碟在这儿

2689: 统计数字字符个数
2711 统计个数
1054 统计字符

课前练习

1.

```
#include<iostream>
using namespace std;
int main()
{
      int  m=65, n=14;
      while (m != n)
      {   while (m > n)  m = m - n;
          while (n > m)  n = n - m;
      }
      printf("m=%d\n",m);

      return 0;
}
```

输出_____m=1_____

2.

```
#include <iostream>
using namespace std;
char a[7],temp;
int i;
int main()
{
    for(i=1; i<=6; i++)
        a[i]=i*2+'A';
    for(i=1; i<=3; i++)
    {
        temp=a[i];
        a[i]=a[7-i];
        a[7-i]=temp;
    }
    for(i=1; i<=6; i++)
        cout<<a[i];
    return 0;
}
```

输出: ___MKIGEC_____

习题

1. 有以下程序：

```
main( )
{  char  w[5]={'a', 'b', 'c', 'd', 'e'};
   int  i;
   for( i=0;i<2;i++ ) w[i]=w[i+2]-32;
   w[i]=w[i]-30;    w[i+1]=w[i+1]-30;
   for( i=0;i<5;i++ ) putchar(w[i]);
}
```

程序运行后的输出结果是(　　B　　)

A) abcde　　B)CDEFe　　C)ABCAB　　D)cdefg

2. 有以下程序：

```
main( )
```

```
{   char a[ ]="book",t;
    int i,k=0;
    for(i=1; i<=3;i++)
       if(a[k]<a[i]) k=i;
    t=a[k];  a[k]=a[3];  a[3]=t;
    puts(a);
}
```

程序运行之后输出的结果是(B)

A)boko B)bkoo C)koob D)book

3.有以下程序：

```
main( )
{   int i,s=1,w;
    char a[ ]="+2468";
    if(a[0]='+')i=1;
    else if( a[0]=='-' )  {  s=-1; i=1; }
    else i=0;
    for ( w=0; a[i]>='0' && a[i]<='9'; i=i+2) w=w*10+a[i]-'0';
    printf("%d\n",s*w);}
```

程序运行后输出的结果是(A)

A)26 B)+26 C)+2468 D)48

4. 以下程序的功能是删除字符串 s 中的所有数字字符。请把适当的选项填在下划线处。

```
main( )
{   int i,j=0; char s[80];
    gets(s);
    for (i=0; s[i]!='\0'; i++)
       if( s[i]<'0' || s[i]>'9' )_____(        )_
    s[j]='\0';
    puts(s);
}
```

A)s[j]=s[i];j++; B)s[++j]=s[i];

C)s[j++]=s[i]; D){j++;s[j]=s[i];}

5. 以下程序段的功能是输出两个字符串中对应字符相等的字符。请把适当的选项填在下划线处。

```
int i=0;  char a[80],b[80];
gets(a); gets(b);
while( a[i]!='\0' && b[i]!='\0' )
{   if( a[i]==b[i] )
      printf("%c",_____________   );
    else  ++i;
}
```

A) a[i] B)b[i] C)b[++i] D)a[i++]

6. 以下程序运行后的输出结果是_6 1___。

```
#include <iostream>
using namespace std;
main( )
{
char  w[5]={ '3','4','2','6','1' },m,n,v;
int  i;
m=w[0];   n=w[0];  v=w[0];
for( i=1;i<5;i++ )
{   if( w[i]>m )  m=w[ i ];
       else if( w[i]<n )  n=w[i];
}
printf("%c   %c",m,n );
}
```

7.以下程序运行后的输出结果是__abc123DEF_____。

```
int main() //输入：ABC123def
{   char ch;
     while((ch=getchar())!='\n')
    { if(ch>='A' && ch<='Z') ch=ch+32;
      else if(ch>='a' && ch<'z') ch=ch-32;
      printf("%c",ch);
    }
    printf("\n");}
```

8.

```
int main()
{
    string a, b;
    int i;
    a = "AABBCCDKKRRSSXX";
    cin >> b;
    for (i = 0; i < b.length(); ++i)
    {
        if ((b[i] >= '0') && (b[i] <= '9'))
            cout << b[i];
        else
        {
            if ((b[i] >= 'A') && (b[i] <= 'Z'))
                cout << a[b[i] - 'A' - 1];
        }
    }
    return 0;
}
```

输入：NOIP-2012　　输出：　SXKX2012

第十一讲 冒泡选择

重点

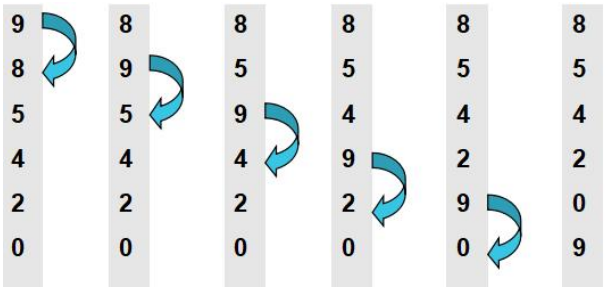
1. 掌握插入排序。
2. 掌握选择排序。

排序的实现基础是数组。所谓排序，是要整理表中的记录，使之按关键字递增（或递减）有序排列，常用的基础排序方法有选择排序、 冒泡排序、 插入排序、 桶排序、 快速排序、 Sort 排序等。

11.1 冒泡排序

冒泡排序是最简单的排序方法之一，该方法的思路是：依次比较相邻的两个数，把大的放前面，小的放后面。即首先比较第 1 个数和第 2 个数，大数放前,小数放后。然后比较第 2 个数和第 3 个数.....直到比较最后两个数。第一趟结束，最小的一定沉到最后。重复上过程,仍从第 1 个数开始,到最后第 2 个数，然后.....

由于在排序过程中总是大数往前，小数往后，相当气泡上升，所以叫冒泡排序。



1. 算法实现

```
for (int i=1;i<=n-1;i++){
    //比较 n-1 次
    for (int j=1;j<=n-i;j++)
    //从一个开始比较，到 n-i 为什么？
     if (a[j]<a[j+1])
     {
        temp=a[j]; a[j]=a[j+1]; a[j+1]=temp;
     }
 }
```

如何下标从 0 开始呢？

```
 for (_____________________){
    //比较 n-1 次
    for (_____________________)
       if (a[j]<a[j+1])
       {
            temp=a[j]; a[j]=a[j+1]; a[j+1]=temp;
```

```
        }
    }
```

1 [2401] 用冒泡法从小到大排序

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[10],n=10;
    for(int i=0;i<10;i++)
        cin>>a[i];
    for(___1_____________________){
        for(__2_____________________){
            if(_3_____________________])
                swap(a[j],a[j+1]);
        }
    }
    for(int i=0;i<n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
    return 0;
}
```

思考：冒泡法是不是一定要比较 n-1 趟？

不一定！比如 8 7 6 5 1 2 3 4 中，n=8，但实际上只需要进行 4 趟比较，后面 3 趟没有进行交换。也就是说，如果在某一趟比较过程中，没有发现前一个数比后一个数大的情况，即没有进行交换数据，那么后面就不需要再进行比较了。

极端的情况，假设 n 个数已经是按从小到大的顺序排好了，那么实际上只需要进行一趟比较就可以得出结论了。

怎么实现？

通过设置状态变量 bchange，为 bool 型，当上一趟有数据交换时，它的值为 ture，否则为 false，它的初始值为 true。在每趟比较之前将这个状态变量置为 false，如果有数据交换，则将 bchange 置为 true，比较完一趟后，判断 bchange 是否为 false，如果仍为 false，则表示这一趟没有进行数据交换，后续的判断就不需再进行了，可以提前退出循环了。

```
#include<bits/stdc++.h>
using namespace std;
#include <stdio.h>
int main( )
{
1   int a[10];
2   int i, j;   //循环变量
3   int t;  //用来实现交换两个数的中间变量
```

```
4   for ( i=0; i<10; i++ )  scanf( "%d", &a[i] ); //输入
5   bool bchange=true;  //状态变量
6   for( j=0; j<9&&bchange; j++)//共进行 9 趟比较
7   {
8       bchange=false;
9       for( i=0; i<9-j; i++)  //在每趟中要进行(10-j)次两两比较
10      {
11        if( a[i]>a[i+1] )
12          {bchange=true; t=a[i]; a[i]=a[i+1]; a[i+1]=t; }
        }
    }
    for( i=0; i<10; i++ )  printf( "%d ", a[i] ); //输出
}
```

11.2 选择法排序

选择排序法的思路：依次从初始序列中选择最小值（最大值）与当前元素做交换，共比较 n-1 趟。

第 1 趟：通过 n-1 次比较，从 n 个数中找出最小的，将它与第 1 个数交换。第 1 趟选择排序完毕，使得最小的数被安置在第 1 个元素位置上。

如，数组 a: 36 25 48 12 65 43 20 58 21 17，第一轮比较：

(1) 假设 k 指向的元素为最小值，初始 k=i(此时 i=1),即默认 a[k]=a[i]=a[1]=36 为最小值。

(2) 接着,a[k]与“25 48 12 65 43 20 58 21 17”中每个元素 a[j]做比较。如果 a[j]<a[k]，则 k=j，一轮比较结束后，a[k]为最小值。

```
for( _____________________ )   //将 a[i+1]~a[n]之间的每个数与 a[k]比较
{
    if(_________________ )  k = j;
}
```

(3) 最后将 a[k]与 a[i]交换。

第 2 趟：通过 n-2 次比较，从剩余的 n-1 个数中找出次小的数，将它与第 1 个数交换。第 1 趟选择排序完毕，使得次小的数被安置在第 1 个元素位置上。

如此重复上述过程，共经过 n-1 趟选择交换后，排序结束。

简单选择排序法也需要用一个二重循环来实现,同样可以带着以下 3 个类似问题来理解其思想(有 n 个数，要求按照从小到大的顺序排序)：

1. 要进行多少趟选择？—— 要进行 n-1 趟选择(第 1 趟，第 2 趟，…，第 n-1 趟)。因此外循环的循环变量 i 的取值是从 1 到 n-1。

2. 在第 i 趟里怎么选？—— 第 i 趟要从 a[i+1], a[i+1], ..., a[n]中选择最小的数，记为 a[k]。i=1，1，2，...，n-1。首先假设 a[i]就是最小的，然后对 a[i+1], ..., a[n-1]每个数都判断一下是否比当前的最小值还要小。因此内循环的循环变量 j 的取值是从 i+1 到 n。

1. 在第 i 趟里怎么交换？—— 每趟选择最终交换的是 a[i]和 a[k]。

```
int i, j, k, t; //t 是用来交换 a[k]和 a[i]的临时变量
for( i=1; i<=n-1; i++ ) //共进行 n-1 趟选择及交换
{
    k = i;      //第 i 趟中最小的数初始为 a[i]
    for( j=i+1; j<=n; j++ ) //将 a[i+1]~a[n]之间的每个数与 a[k]比较
    {
        if( a[ j ] < a[ k ] )  k = j;
    }
    t = a[ i ];  a[ i ] = a[ k ];  a[ k ] = t;  // 交 换 a[k] 与 a[i] ， 或 用
    swap(a[i],a[k])
}
```

2 [2495] 选择排序

```
#include<iostream>
using namespace std;
int main(){

    int t,n,a[1005],i,j,k,p;
    cin>>t;
    for(p=1;p<=t;p++){
        cin>>n;
        for(i=1;i<=n;i++)
            cin>>a[i];
        for(i=1;i<n;i++){
            _1_____________
            for(j=_2_____________)
                if(_3_____________)k=j;
            swap(_4_____________);
        }
        for(i=1;i<=n;i++)
            cout<<a[i]<<" ";
        cout<<endl;
    }
    return 0;
}
```

简单选择法的改进：在第 i 趟选择完毕，如果当前最小的数就是 a[i]，即 k = i，则不需要交换 a[k]和 a[i]。即在上述程序中，交换 a[k]和 a[i]的 3 条语句前可以加上一个条件判断。代码如下：

```
if( k!=i )
{  t=a[k];  a[k]=a[i];  a[i]=t;  }      //交换 a[k]和 a[i]
```

11.3 案例讲解

3 [2225] 基础排序

输入一个正整数 n(1<n<=10)，再输入 n 个整数，将它们从大到小排序后输出。
输入
4
5　1　7　6
输出
7　6　5　1

冒泡排序：

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
        int a[11];
        int n,k,t,i;
        cin>>n;
        for(i=0;i<n;i++)
                cin>>a[i];
```

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

4 *[3335] 陶陶摘苹果

题目描述

又是一年秋季时，陶陶家的苹果树结了 n 个果子。陶陶又跑去摘苹果，这次她有一个 a 公分的椅子。当他手够不着时，他会站到椅子上再试试。这次与原来的摘苹果不同的是：陶陶之前搬凳子，力气只剩下 s 了。当然，每次摘苹果时都要用一定的力气。**陶陶想知道在 s<0 之前最多能摘到多少个苹果。**

现在已知 n 个苹果到达地上的高度 xi，椅子的高度 a，陶陶手伸直的最大长度 b，陶陶所剩的力气 s，陶陶摘一个苹果需要的力气 yi，求陶陶最多能摘到多少个苹果。

所有数据：n<=5000 a<=50 b<=200 s<=1000
输入
第 1 行：两个数 苹果数 n，力气 s。
第 2 行：两个数 椅子的高度 a，陶陶手伸直的最大长度 b。
第 3 行~第 3+n-1 行：每行两个数 苹果高度 xi，摘这个苹果需要的力气 yi。
输出
只有一个整数，表示陶陶最多能摘到的苹果数。
样例输入
8 15
20 130
120 3
150 2
110 7
180 1
50 8
200 0
140 3
120 2
样例输出
4

分析：为了能摘到最多的苹果，陶陶可以先摘需要力气最少的苹果，即按力气从小到大排序，排完序后，按力气从小到大选择能摘到的苹果即可。

```
#include<bits/stdc++.h>
#include<iostream>
using namespace std;
int main()
{
    int n,s,xi[100]={0},yi[100]={0},a,b,sum=0,z,y;
    cin>>n>>s;//果数 n，力气 s。
    cin>>a>>b;//椅子的高度 a，陶陶手伸直的最大长度 b。
    a+=b;//最终高度
    for(int i=1;i<=n;i++)
        cin>>xi[i]>>yi[i];// 苹果高度 xi，摘这个苹果需要的力气 yi。
    for(int i=1;i<n;i++)
    {
        for(int j= 1                    )
        {//排序，按照果需要的力气 yi 排序
            if( 2                  )
            {//同时要交换苹果高度
                y=xi[i];xi[i]=xi[j];xi[j]=y;
                z=yi[i];yi[i]=yi[j];yi[j]=z;
            }
```

```
        }
    }
    for(int i=1;i<=n;i++)
    {
        if(__3_____________)
        {//按照力气从小到大取苹果，直到力气用完
            sum++;
            4
        }
    }
    cout<<sum<<endl;
    return 0;
}
```

5 *[2744] 合影效果

小云和朋友们去爬香山，为美丽的景色所陶醉，想合影留念。如果他们站成一排，男生全部在左（从拍照者的角度），并按照从矮到高的顺序从左到右排，女生全部在右，并按照从高到矮的顺序从左到右排，请问他们合影的效果是什么样的（所有人的身高都不同）？

输入

第一行是人数 n（2 ≤ n ≤ 40，且至少有 1 个男生和 1 个女生）。

后面紧跟 n 行，每行输入一个人的性别（男 male 或女 female）和身高（浮点数，单位米），两个数据之间以空格分隔。

输出

n 个浮点数，模拟站好队后，拍照者眼中从左到右每个人的身高。每个浮点数需保留到小数点后 2 位，相邻两个数之间用单个空格隔开。

样例输入

6
male 1.72
male 1.78
female 1.61
male 1.65
female 1.70
female 1.56

样例输出

1.65 1.72 1.78 1.70 1.61 1.56

思路：先生成男生数组和女生数组，然后对两个数组做排序即可。

```
#include <iostream>
#include <iomanip>
using namespace std;
const int LN=40;
```

```
double boy[LN],girl[LN];
int main(){
    int n;
    cin>>n;
    int aln=0,bln=0;
    string s;//定义一个字符串
    double h;
    for(int i=0;i<n;i++){
        cin>>s>>h;//进行字符串比较确定是男生还是女生
        if(s=="male") _1_____ boy[aln++]=h;//加入男生队伍
        else  _2_____girl[bln++]=h;//加入女生队伍
    }
    for(int i=0;i<aln;i++){
        int k = _3_____  i;//选择排序
        for(int j=i+1;j<aln;j++)
            if(_4_____ boy[j]<boy[k])_5_____    k=j;
        if(k!=i) swap(boy[i],boy[k]);
    }
    for(int i=0;i<bln;i++){
        int k =i;
        for(int j=i+1;j<bln;j++)
            if(girl[j]>girl[k]) k=j;
        if(k!=i) swap(girl[i],girl[k]);
    }

    for(int i=0;i<aln;i++)
        printf("%.2lf ",boy[i]);
    for(int i=0;i<bln;i++)
        printf("%.2lf ",girl[i]);
    return 0;
}
```

课堂作业列表

[2401] 用冒泡法从小到大排序
[1938]排序
[2225] 基础排序
[3335] 陶陶摘苹果
2748 明明的随机数
2750 出现次数超过一半的数
2746 病人排队

课前练习

1 #include <iostream>	2 #include<iostream>

```
using namespace std;
char a[7],temp;
int i;
int main()
{
    for(i=1; i<=6; i++)
        a[i]=i*2+'A';
    for(i=1; i<=3; i++)
    {
        temp=a[i];
        a[i]=a[7-i];
        a[7-i]=temp;
    }
    for(i=1; i<=6; i++)
        cout<<a[i];
    return 0;
}
```

输出:___MKIGEC_____

```
using namespace std;
int main( )
{
    char a[8],temp; int j,k;
    for(j=0;j<7;j++) a[j]='a'+j;
    a[7]='\0';
    for(j=0;j<3;j++)
    {
       temp=a[6];
       for(k=6;k>0;k--) a[k]=a[k-1];
       a[0]=temp;
       printf("%s\n",a);
    }
}
```

以下程序运行后的输出结果是_____

gabcdef

fgabcde

efgabcd

习题

1. 若用冒泡排序法对序列{10,14,26,29,41,52}从大到小排序，需进行（　　）次比较。

A 25　　B. C　　C. 15　　D. 10

2. 对于 7 个数进行冒泡排序，需要进行的比较次数为：（　　）

A. 7　　B. 14　　C. 21　　D. 49

3. 对于 10 个数的简单选择排序，最坏情况下需要交换元素的次数为：（　）

A.9　　B.36　　C.45　　D.100

4.对 n 个不同的排序码进行冒泡排序,在元素无序的情况下比较的次数最多为( )。

A.n+1　　B.n　　C.n-1　　D.n(n-1)/2

5.若用冒泡排序方法对序列{10,14,26,29,41,52}从大到小排序，需进行 （ ）次比较。

A.3　　B.10　　C.15　　D.25

6.输入 10 个数据，进行冒泡排序

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int a[10];
    int n=10;
    for(int i=0;i<10;i++)
        cin>>a[i];
    for(int i=0;_____1_____;i++){
        for(int j=0;____2_____;j++){
            if(___3_____)
```

```
                _______4________;
        }
    }
    for(int i=0;i<n;i++)
        cout<<a[i]<<" ";
    cout<<endl;
    return 0;
}
```

7. 选择排序

```
#include<iostream>
using namespace std;
int main(){

    int t,n,a[1005],i,j,k,p;
    cin>>t;
    for(p=1;p<=t;p++){
        cin>>n;
        for(i=1;i<=n;i++)
            cin>>a[i];
        for(i=1;i<n;i++){
            _1_________________
            for(j=_2_________________)
                if(_3_________________)k=j;
            swap(_4_________________);
        }
        for(i=1;i<=n;i++)
            cout<<a[i]<<" ";
        cout<<endl;
    }
    return 0;
}
```

8. 阅读程序	9 阅读程序

#include <iostream>
#include <string>
using namespace std;
int main()
{
int i,j,n,n1;
cin>>n;
i=2;
	#include <iostream>
	#include <cmath>
	using namespace std;
	int main()
	{
	int n, x, i, temp, j, count=0;
	cin >> n >> x;
	for (i = 1; i <= n; i++)

```
    j=0;
    n1=n;
    while(n1!=1)
    {
        while(n1%i==0)
        {
            j++;
            if(j==1)
                cout<<n<<"="<<i;
            else
                cout<<"*"<<i;
            n1=n1/i;
        }
        i++;
    }
    return 0;
}
```

输入：

102

输出：102=2*3*17

```
    {
        temp = i;
        while (temp > 0)
        {
            j = temp % 10;
            temp = temp / 10;
            if (j == x)
                count++;
        }
    }
    cout << count << endl;
    return 0;
}
```

输入：

100 5

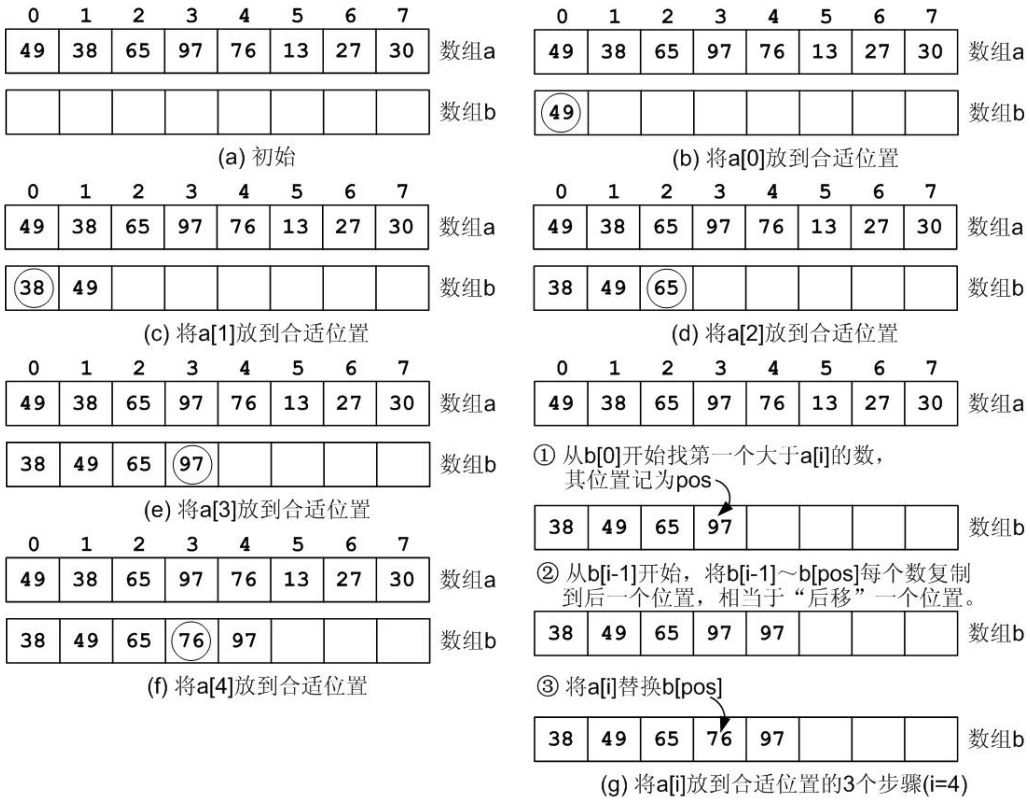
输出：20

第十二讲 插入排序

重点

1. 掌握选择排序。
2. 理解 sort 排序。

插入排序是一种简单的排序方法，其算法的基本思想是：假设待排序的数据存放在数组 a[1..n]中。例如:设 n=8，数组 a 中 8 个元素是: 49, 38, 65, 97, 76, 13, 27, 30，这 8 个数已经存储在数组 a 了，把这 8 个数依次插入到数组 b 中。



12.1 两个数组顺插

如果第一个元素为 a[0]，目前***待插入的数是 a[i]***，这时数组 b 中的 b[0]~b[i-1]已经有序了，将 a[i]插入到数组 b 中的合适位置；插入 a[i]的过程，分为 3 个步骤：

① 从 b[0]开始，在 b[0]~b[i-1]范围内找第一个大于 a[i]的数，其位置记为 pos；

```
for(j=0;j<=i-1;j++)
    if(b[j]>a[i])break;
pos=j;
```

② 从 b[i-1]开始，将 b[i-1]~b[pos]复制到后一个位置，相当于将这些数“后移”一个位置；

```
for(k=n-1;k>=pos;k--)
    b[k+1]=b[k];
```

③ 将a[i]放置在b[pos]位置上，替换b[pos]。

```
b[pos]=a[i];
```

1 [1938]排序

插入法实现对输入的10个数按从小到大进行排序。

```
#include<bits/stdc++.h>
using namespace std;
int main( )
{
    int a[10], b[10];   //输入的10个数及排序后的10个数
    int i, j, k,n=10;   //循环变量
    int pos;//每个数的插入位置
    for ( i=0; i<n; i++ )  scanf( "%d", &a[i] ); //输入
    for( i=0; i<n; i++ )//将a[i]插入合适的位置
    {
        //在数组b中，b[0]~b[i-1]已经有序了，给a[i]找到合适的位置
        for( j=0; j<=i-1; j++ )
            if( b[j]>a[i] )  break;
        pos = j;    //1 找到a[i]要插入的位置

        for( k=i-1; k>=pos; k-- )   // 2 将b[pos]~b[i-1]后移一个位置
            b[k+1] = b[k];

        b[pos] = a[i];  //3 将a[i]放置在b[pos]位置上
    }
    for( i=0; i<10; i++ )  printf( "%d ", b[i] ); //输出
    printf( "\n" );
}
```

填空

```
#include<bits/stdc++.h>
using namespace std;
int main( )
{
    int a[10], b[10];   //输入的10个数及排序后的10个数
    int i, j, k,n=10;   //循环变量
    int pos;    //每个数的插入位置
    for ( i=0; i<n; i++ )  scanf( "%d", &a[i] ); //输入
    for( i=0; i<n; i++ )
    {

        for( j=0;_1_____________)
            if(_2___________ )  break;
```

```
        __3_____________    //a[i]的插入位置

        for( k=_4_____________ )   //将 b[pos]~b[i-1]后移一个位置
            b[_5_____] = b[_____];

        b[_6____] = a[___]; //将 a[i]放置在 b[pos]位置上
    }
    for( i=0; i<10; i++ )  printf( "%d ", b[i] ); //输出
    printf( "\n" );
}
```

12.2 一个数组顺插

能否在用一个数组，完成插入排序呢？

假设数组 a[10]: 36 25 48 12 65 43 20 58 21 17 ；**从第二个元素 a[i]开始，依次把 a[i]插入到前面已排好的序列,在插入过程中需要先保存 a[i],因为在插入后移过程中 a[i]会被替换。** 执行插入排序的过程如下，其数据变动情况：

第 1 步:[36] 25 48 12 65 43 20 58
第 2 步:[25 36] 48 12 65 43 20 58
第 3 步:[25 36 48] 12 65 43 20 58
第 4 步:[12 25 36 48] 65 43 20 58
第 5 步:[12 25 36 48 65] 43 20 58
第 6 步:[12 25 36 43 48 65] 20 58
第 7 步:[12 20 25 36 43 48 65] 58
第 8 步:[12 20 25 36 43 48 58 65]

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    //插入            43 a[i]
    //已排好的序列为  12 25 36 48 65
   1 int a[10],n=10,t;
   2 int i,j,k,pos;
   3 for(i=0;i<n;i++)cin>>a[i];
   4 for(i=1;i<n;i++){
   5    t=a[i];//因为在原数组上挪动，a[i]会被替换，所以要保存
   6     for(j=0;j<=i-1;j++)
   7         if(t<a[j])break;//1 从前往后找到第一个比 a[i]大的数
   8     pos=j;//2 记录位置
   9     for(k=i-1;k>=pos;k--)
   10         a[k+1]=a[k]; //3 移动
   11    a[pos]=t; //4 插入
    }
```

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

    return 0;
}
```

12.3 一个数组倒插

[12 25 36 48 65] 43 20 58 21 17

步骤：
1 从后往前扫描，比较当前数 t=a[i],与 a[j]的值。
2 如果 t<a[j](如 a[j]=65，t=43,)，则 a[j]往后挪，此时会替换 43。
3 继续往前找，如果 t>=a[j],则在失配的位置插入 t。

```
#include<bits/stdc++.h>
using namespace std;
int main(){
1    int a[10],n=10,t;
2    int i,j,k,pos;
3    for(i=0;i<n;i++) cin>>a[i];
4    for(i=1;i<n;i++){
5       t=a[i];//记录当前要插入的元素
6       for(j=i-1;j>=0&&t<a[j];j--)//从后往前，一边比较一边挪，
7           a[j+1]=a[j];
8      a[j+1]=t;
9   }
    for(i=0;i<n;i++)
        cout<<a[i]<<" ";
    return 0;
}
```

填空：

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    int a[10],n=10,t;
    int i,j,k,pos;
    for(i=0;i<n;i++)
        cin>>a[i];
    for(i=1;i<n;i++){
       t=__1__________
       for(j=_2_____________________)
            a[_3_______]=a[_3_______];
```

```
        a[_4_________]=t;
    }
    for(i=0;i<n;i++)
        cout<<a[i]<<" ";

    return 0;
}
```

12.3 Sort 排序

使用：#include <algorithm>
        using namespace std;
作用：排序
时间复杂度：n*lg(n)

实现原理：**sort 并不是简单的快速排序，它对普通的快速排序进行了优化，此外，它还结合了插入排序和堆排序**。系统会根据你的数据形式和数据量自动选择合适的排序方法，这并不是说它每次排序只选择一种方法，它是在一次完整排序中不同的情况选用不同方法，比如给一个数据量较大的数组排序，开始采用快速排序，分段递归，分段之后每一段的数据量达到一个较小值后它就不继续往下递归，而是选择插入排序，如果递归的太深，他会选择堆排序。

形式：sort(first_pointer,first_pointer+n,cmp)
参数解释：

第一个参数是数组的首地址，一般写上数组名就可以，因为数组名是一个指针常量。

第二个参数相对较好理解，即**首地址加上数组的长度 n**（代表尾地址的下一地址）。

最后一个参数是比较函数的名称（自定义函数 cmp），这个比较函数可以不写，即第三个参数可以缺省，这样 sort 会默认按数组升序排序。

简单例子：对数组 a 的 0~n-1 元素进行升序排序，只要写 sort(a,a+n)即可。

sort 不只是能像上面那样简单的使用，我们可以对 sort 进行扩展，关键就在于第三个参数<cmp 比较函数>，我们想降序排列，或者说我不是一个简简单单的数组，而是结构体、类怎么办，下面给出一些方法和例子。

```
 //情况一：数组排列
int A[100];
bool cmp(int a,int b)//int 为数组数据类型
{
    return a>b;//降序排列
    //return a<b;//默认的升序排列
}
sort(A,A+100,cmp);
```

12.4 案例讲解

2 [2225] 基础排序

输入一个正整数 n(1<n<=10)，再输入 n 个整数，将它们从大到小排序后输出。
输入
4
5　1　7　6
输出
7　6　5　1

```
#include<bits/stdc++.h>
using namespace std;
int cmp(int a,int b){
        return a>b;
}
int main(){
        int a[11];
        int i,n;
        cin>>n;
        for(i=0;i<n;i++)
                cin>>a[i];
        sort(a,a+n,cmp);
        for(i=0;i<n;i++)
                cout<<a[i]<<"  ";
        return 0;
}
```

3 [2736] 奇数单增序列

给定一个长度为 N（不大于 500）的正整数序列，请将其中的所有奇数取出，并按升序输出。

输入第 1 行为 N；第 2 行为 N 个正整数，其间用空格间隔。

输出增序输出的奇数序列，数据之间以逗号间隔。数据保证至少有一个奇数。

样例输入
10
1 3 2 6 5 4 9 8 7 10
样例输出
1,3,5,7,9

```
#include<iostream>
```

```
#include<algorithm>
using namespace std;
 int a[555];
int main(){
    int n,m,i;
    cin>>n;
    for(i=1;i<=n;i++)
         cin>>a[i];
    ____________________________//排序 sort(a+1,a+1+n);
    int q=0;//记录输出状态
    for(i=1;i<=n;i++)
    {
        if(__________________ a[i]%2==1)//如果是奇数
        {
            if(q++)cout<<",";
            cout<<a[i];
        }
    }
    cout<<endl;
    return 0;
}
```

4 [2228] 部分排序

输入一个正整数 n(5<n<=15)，再输入 n 个整数，将后面 5 个数排序（从小到大）后输出。
样例输入
6
6 5 4 3 2 1
样例输出
6 1 2 3 4 5

```
#include<bits/stdc++.h>
using namespace std;
int main ()
{
        int n;
        int a[20];
        cin>>n;
        for(int i=1;i<=n;i++)
            cin>>a[i];
        sort(a+n-4,a+n+1);
        for(int i=1;i<=n;i++)
            cout<<a[i]<<" ";
        return 0;
```

}

5 *[2744] 合影效果

小云和朋友们去爬香山，为美丽的景色所陶醉，想合影留念。如果他们站成一排，男生全部在左（从拍照者的角度），并按照从矮到高的顺序从左到右排，女生全部在右，并按照从高到矮的顺序从左到右排，请问他们合影的效果是什么样的（所有人的身高都不同）？
输入
第一行是人数 n（2 ≤ n ≤ 40，且至少有 1 个男生和 1 个女生）。
后面紧跟 n 行，每行输入一个人的性别（男 male 或女 female）和身高（浮点数，单位米），两个数据之间以空格分隔。
输出
n 个浮点数，模拟站好队后，拍照者眼中从左到右每个人的身高。每个浮点数需保留到小数点后 2 位，相邻两个数之间用单个空格隔开。

样例输入
6
male 1.72
male 1.78
female 1.61
male 1.65
female 1.70
female 1.56
样例输出
1.65 1.72 1.78 1.70 1.61 1.56

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

int cmp(double a,double b)
{
    return a>b;
}
int main()
{
    double a[41],b[41];
    int n,x=0,y=0;
    string s;
    double h;
    cin>>n;
    for(int i=0;i<n;i++)
    {
        cin>>s>>h;
        if(s=="male")
```

```
            a[y++]=h;
        else
            b[x++]=h;
    }
    sort(a,a+y);
    sort(b,b+x,cmp);
    for(int i=0;i<y;i++)
        printf("%.2lf ",a[i]);
    for(int i=0;i<x;i++)
        printf("%.2lf ",b[i]);
    return 0;
}
```

课堂作业列表

[2495] 选择排序	[7614] 统计同成绩学生人数
[2677] 去重	[2225] 基础排序
[2750] 出现次数超过一半的数	[2496] 排序
[2736] 奇数单增序列	[2228] 部分排序

课前练习

1.

输出：_____

```
int main(){
        char str[1000];
        int len;
        int i;
        gets(str);
        len=strlen(str);
        for(i=0;i<len;i++){
                if(str[i]>='a'&&str[i]<='z')
                        str[i]-=32;
        }
        puts(str);
        return 0;
}
```

输入：helloworld123Ha

输出：HELLOWORLD123HA

2.

```
int main()
{
    int x, y, t;
    cin >> x >> y;
    if (x > y)
    {
        t = x;x = y;y = t;
    }
    t = 0;
    while (y >= 0)
    {
        y = y - x;
        t = t + 1;
    }
    cout << "t=" << t - 1;
    return 0;
}
```

输入：

512 16

输出： t=32

习题

1. 用直接插入排序方法对下面四个序列进行排序（由小到大），元素比较次数最少的是（　　）。

A. 94,32,40,90,80,46,21,69

B. 21,32,46,40,80,69,90,94

C. 90,69,80,46,21,32,94,40

D. 32,40,21,46,69,94,90,80

2. 在对一组关键字序列{70，55，100，15，33，65，50，40，95}，进行直接插入排序时，把 65 插入，需要比较(　　)次。

A. 2　　B. 3　　C. 4　　D. 5

3. 有以下程序

```
int main()
{ int s[12]={1,2,3,4,4,3,2,1,1,1,2,3},c[5]={0},i;
   for(i=0;i<12;i++) c[s[i]]++;
   for(i=1;i<5;i++) printf(" %d",c[i]);
   printf("\n");
}
```

程序的运行结果是 (C　　)

A)1 2 3 4　　B)2 3 4 4　　C)4 3 3 2　　D)1 1 2 3

4.读程序写结果_____1　0　2　2　5　7　13　20_____

```
#include <stdio.h>
void main()
{
        int a[8]={1,0,1,0,1,0,1,0},i;
        for(i=2;i<8;i++)
            a[i]+= a[i-1] + a[i-2];
        for(i=0;i<8;i++)
            printf(“%5d”,a[i]);
}
```

5. 直接插入排序法填空：

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
        int t,a[100],b[100],i,j,k;
        int n=10,pos;
        for(i=0;i<n;i++)
                cin>>a[i];

        for(i=0;i<n;i++){
                for(_____1_____j++)
                        if(b[j]>a[i])break;
```

```
            __________2__________//第一步    找到位置
            for(k=i;k>=pos;k--)
                    _________3____________//第二步 移动数据

          ______4________;//第三步 插入
        }
        for(i=0;i<n;i++)
                cout<<b[i]<<" ";
            return 0;}
```

6 阅读程序

```
#include <iostream>
using namespace std;
int main()
{
    int m, n, i, j, s;
    int d[101];
    cin >> n;
    for (m = 10; m <= n; m++)
    {
        s = m * m;
        j = 0;
        while (s > 0)
        {
            j = j + 1;
            d[j] = s % 10;
            s = s / 10;
        }
        i = 1;
        while (d[i] == d[j] && (i < j))
        {
            i = i + 1;
            j = j - 1;
        }
        if (i >= j)
            cout << m << endl;
    }
    return 0;
}
```

输入：

30

输出： _____

```
11
22
26
```

7.
getline(cin,s):接收一个字符串 s，且可以接收空格；
str1.compare(str2):比较两个字符串的大小，如果相等，输出 0，不等输出-1。

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s,s1;
    int i,a,b,k;
    getline(cin,s);
    s1="you";
    k=0;
    a=s.length();
    b=s1.length();
    for(i=0; i<=a-b+1; i++)
    {
        if(s1.compare(s.substr(i,b))==0)
            k=k+1;
    }
    cout<<k<<endl;

    return 0;
}
```

输入：It’s easy. You get off the bus. Then you cross the road. You take the first road on the left. You walk for five minutes.
输出：_____1_____

第十三讲 桶与状态

桶排序是状态变量的应用之一，其思想是若待排序的记录的关键字在一个明显有限范围内(整型，如 x<1000000)时，可设计有限个有序桶，**每个桶装入一个值**（当然也可以装入若干个值），顺序输出各桶的值，将得到有序的序列。

例：输入 n 个 0 到 100 之间的不相同整数，由小到大排序输出。

```
memset(b,0,sizeof(b));  //初始化
for( i=1;i<=n;i++)
{
    cin>>k;
    b[k]=1; //将关键字等于 k 的值全部装入第 k 桶
}
for( i=0; i<=100;i++)
    while (b[i]>0)  {
        cout<<i<<" " ;
        b[i]--;
    }    //输出排序结果
}
```

桶排序代码：

```
#include<iostream>
#include <cstring>
using namespace std;
int main()
{
    int b[101],k,i,n;
    memset(b,0,sizeof(b));                    //初始化
    cin>>n;
    for( i=1;i<=n;i++)
    {
        cin>>k;
        _1_____________________    //将关键字等于 k 的值全部装入第 k 桶
    }
    for( i=0; i<=100;i++)
       while (_2_____________)  {
            cout<<i<<" " ;
        __3_____________________
        }    //输出排序结果
    cout<<endl;
}
```

1 [7799] 统计

输入 n(n<=1000)个数（每个数保证在 5 以内）,输出 0~5 中，每个数字出现的次数
输入两行 第一行 n 第二行 n 个数
输出
按顺序输出每个数的出现次数，一行一个数。如果没有出现过，则输出 0。 对于例子中的数组中的数是在 5 以内（包括 5），因此我们只统计{0,1,2,3,4,5}的出现频数。
样例输入 复制

```
6
1 1 1 0 3 4
```

样例输出 复制

```
1
3
0
1
1
0
```

```
#include <iostream>
using namespace std;
int main()
{
   int a[6],b[6]={0},s=0,n;
   cin>>n;
   for(int k=1;k<=n;k++){
      cin>>a[k];
    b[a[k]]++;
   }
   for(int i=0;i<=5;i++)cout<<b[i]<<endl;
   return 0;
}
```

2 [2677] 去重

给定含有 n 个整数的序列，要求对这个序列进行去重操作。所谓去重，是指对这个序列中每个重复出现的数，只保留该数第一次出现的位置，删除其余位置。

输入包含两行：第一行包含一个正整数 n（1 ≤ n ≤ 20000），表示第二行序列中数字的个数；第二行包含 n 个整数，整数之间以一个空格分开。每个整数大于等于 10、小于等于 5000。

输出只有一行，按照输入的顺序输出其中不重复的数字，整数之间用一个空格分开。
样例输入

```
5
10 12 93 12 75
```

样例输出

10 12 93 75

方法一：从第一个元素 a[i]开始，从前往后扫描，把 a[i+1 ,n]的元素与 a[i]作比较，如果后面的数与 a[i]相等就做标记，最后把没有做标记的数字输出。

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n,i,j,a[20005];
    cin>>n;
    for(i=0;i<n;i++){
        cin>>a[i];
    }
    for(i=0;i<n;i++){//
        for(j=__1_____________){//从前往后扫描 j=i+1;j<n;j++
            if(a[j]==_2____)a[j]=_3_____;//如果后面的数与 a[i]相等就做标
                                          if(a[j]==a[i])a[j]=0;
        }
    }
    for(int i=0;i<n;i++){
         if(a[i]_4______){//(a[i]!=0)
            cout<<a[i]<<" ";
        }
    }
}
```

方法二：谈谈以下去重方法的思路：

_____

_____

```
#include<iostream>
using namespace std;
int main()
{
    int a[20020];
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        cin>>a[i];
        for(int j=1;j<i;j++)
        {
            if(a[j]==a[i])
            {
```

```
                i=i-1;
                n=n-1;
            }
        }
    }
    for(int i=1;i<=n;i++)
    {
        cout<<a[i]<<" ";
    }
    return 0;
}
```

桶排序去重：把元素 a[i]，放置到桶 b 中，即输入一个数，就记为 b[a[i]]=1，这样凡是出现过的数 b[a[i]]=1。然后按照输入的序列，判断 b[a[i]]是否等于 1，如果是 1 就输出 a[i].

```
#include<iostream>
using namespace std;
int n,a[20005],b[5005];
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++)cin>>a[i];
    for(int i=0;i<=5000;i++)b[i]=0;//桶的初始化
    for(int i=1;i<=n;i++)
    {
        if(b[_1_____]==0)//如果桶的值为 1，则这个数是第一次出现直接输出
        {
           cout<<a[i]<<" ";
           b[__2_____]=1;//把该数记为已经出现过
        }
    }
    return 0;
}
```

3 [2750] 出现次数超过一半的数

给出一个含有 n（0 ＜ n ≤ 1000）个整数的数组，请找出其中出现次数超过一半的数。数组中的数大于 0 且小于 10000。

输入
第一行包含一个整数 n，表示数组大小；
第二行包含 n 个整数，分别是数组中的每个元素，相邻两个元素之间用单个空格隔开。
输出
如果存在这样的数，输出这个数；否则输出 no。

样例输入
3
1 2 2
样例输出 2

分析：统计每个数 a[i]出现的次数 b[a[i]]，再比较 s 是否 b[a[i]]>n/2。

```
#include<iostream>
using namespace std;
int main()
{
        int a[1005]={0},b[10005]={0},i,n;
        int flag=1;//没找到
        scanf("%d",&n);
        for(i=1;i<=n;i++)scanf("%d",&a[i]);
        for(i=1;i<=n;i++){
                ____1_____________//元素 a[i]出现的次数 b[a[i]]++;
        }
        for(i=1;i<=n;i++)
                if(_2_____________>n/2){//b[a[i]]
                        printf("%d",a[i]);
                        flag=0; //找到了
                        break;
                }
        if(flag==1)printf("no\n");
        return 0;
}
```

4 [3758] 分数统计任务

输入一些学生的分数，哪个分数出现的次数最多？如果有多个并列，从小到大输出。

任务 1：分数均为不超过 100 的非负整数。

输入

第一行正整数 n，不超过 10000；

第二行 n 个不超过 100 的非负整数。

输出

多行，出现次数最多的分数和次数，有多个分数出现次数相同时，从小到大输出，一行一个。

样例输入

10

89 98 99 89 99 76 87 88 86 77

样例输出

89 2

99 2

```
#include<iostream>
```

```
using namespace std;//3758 分数统计任务 1
int main(){
   int a[10005],n,b[105]={0};
   int maxn=0;
   cin>>n;
   for(int i=1;i<=n;i++){
      __1_____________________  //89 98 99 89 99 76 87 88 86 77
     ___2_____________________ //b[89]++ b[98]++ b[89]++
   }
   for(int i=0;i<=100;i++)
      __3_____________________//求出出现次数最多的数
   for(int i=0;i<=100;i++)
      if(b[i]==maxn) cout<<i<<" "<<maxn<<endl;

   return 0;
}
```

5 [2690] 找第一个只出现一次的字符

题目描述
给定一个只包含小写字母的字符串，请你找到第一个仅出现一次的字符。如果没有，输出 no。

输入一个字符串，长度小于 100。
输出输出第一个仅出现一次的字符，若没有则输出 no。

样例输入
abcabd
样例输出
C

方法 1：用一个数组存储每一个字符出现的次数，求好后，从头扫描，如果只出现 1 次就输出。

```
#include<bits/stdc++.h>
#include<bits/stdc++.h>
using namespace std;
int main( ){
    char a[100];
    int =__1_____________________;//数组 b 表示每个字符出现的次数
    gets(a);
    for(___2_____________________)//
     __3_____________________;// 把字符转化为 数组的下标 a->0 b->1  c->2,
    for(int i=0;i<strlen(a);i++)
     if(__4_____________________)//寻找第一次出现的字符
```

```
     {
        cout<<a[i]<<endl;
        flag=0;
         break;
     }
     if(flag==1)
     cout<<"no"<<endl;
    return 0;
}
```

方法二：从头到尾，对对每个字符进行扫描，看看有没有跟它相等的字符，如果没有则输出该字符。

```
#include<iostream>
#include<cstring>
using namespace std;
int main()
{
    char a[103],st;
    int n,flag=1;
    gets(a);
    n=strlen(a);
    for(int i=0;i<n;i++)
    {//对每个字符进行扫描，看看有没有跟它相等的字符
        st=_1_________;//记录当前的字符
        __2___________;//flag 表示默认当前字符就是第一个出现 1 次的字符
        for(int j=0;j<n;j++)
        {//当前字符为 a[i]，a[i]跟其他所有字符比较
            if(_3_____________)
            {//出现其他字符跟它相等的字符
                flag=0;//不是
                _4_______//退出当前循环，查找下一个
            }
        }
        if(_5_______)
        {//是
            cout<<st<<endl;
            return 0;
        }

    }
    cout<<"no"<<endl;
}
```

6 *[3328]珠心算测验

珠心算是一种通过在脑中模拟算盘变化来完成快速运算的一种计算技术。珠心算训练，既能够开发智力，又能够为日常生活带来很多便利，因而在很多学校得到普及。

某学校的珠心算老师采用一种快速考察珠心算加法能力的测验方法。他随机生成一个正整数集合，集合中的数各不相同，然后要求学生回答：其中**有多少个数，恰好等于集合中另外两个（不同的）数之和**？

最近老师出了一些测验题，请你帮忙求出答案。

输入

共两行，第一行包含一个整数 n，表示测试题中给出的正整数个数。

第二行有 n 个正整数，每两个正整数之间用一个空格隔开，表示测试题中给出的正整数。

输出

一个整数，表示测验题答案。

15

100 12 88 44 46 2 56 99 23 32 11 47 56 9991 9993

8

样例输入

4

1 2 3 4

样例输出

2

思路：_____

_____

```
#include<iostream>
using namespace std;
int t[200005],g[200005];//t 是桶，t[i]表示值为 i 的数在集合中两两相加出现了几次，
g[i]表示值为 i 的数是否在集合中，1 为在，0 为不在
int n,a[105],ans;
int main(){
        cin>>n;
        for (int i=1;i<=n;i++){
                cin>>a[i];//读入
                g[a[i]]=1;//在集合中赋值为 1
        }
        for (int i=1;i<n;i++){//枚举
                for (int j=i+1;j<=n;j++){
                        t[a[i]+a[j]]++;//被加出来了
                }
        }
        for (int i=1;i<=200002;i++){
                if (t[i]>0&&g[i]) ans++;//判断是否满足，满足 ans++
```

```
        }
        cout<<ans<<endl;
        return 0;
}
```

7 *[1609] 素数-大数据

素数是这样的整数，它除了能表示为它自己和 1 的乘积以外，不能表示为任何其它两个整数的乘积。例如，15＝3*5，所以 15 不是素数。又如，12＝6*2＝4*3，所以 12 也不是素数。另一方面，13 除了等于 13*1 以外，不能表示为其它任何两个整数的乘积，所以 13 是一个素数。 你的任务是计算出所有小于等于给定正整数的素数个数(我们认为 1 不是素数)。

输入输入文件中包含多个测试数据。每个测试数据占一行，为一个正整数 N(0<N≤10000000)，即给定正整数。测试数据一直到文件尾。

输出对输入文件中的每个正整数，输出所有小于等于给定它的素数个数。

样例输入	样例输出
2	1
3	2
4	2
5	3
9	4

```
#include<bits/stdc++.h>
using namespace std;
int main()
{//先把素数打表存在 数组 a 中
    int a[10000005]={0};// a[i]=0 表示 i 是素数，初始每个数都是素数
    int n,x,p,j;
    n=int(sqrt(double(10000000)))+1;// 确定倍数的上限值
    for(int i=2;i<=n;i++)
    {//把 i 的所有倍数都设置为非素数即，a[i]=1
        if(a[i]==1)continue;//
        for(j=2*i;j<=10000000;j=j+i)
        {//把所有是 i 的倍数的 j 都设置为 1，即不是素数
            if(a[j]!=1)a[j]=1;
        }
    }
    while(cin>>p)
    {
        x=0;
        for(int i=2;i<=p;i++)
            if(a[i]==0)x++;
        cout<<x<<endl;
    }
```

```
    return 0;
}
```

课前练习

```
#include <iostream>
using namespace std;
int main()
{
    int ss[1001];
    int i,k;
    for(i=1; i<=20; i++)
        ss[i]=i;
    ss[1]=0;
    i=2;
    while(i<=20)
    {
        k=1;
        while(k+i<=20)
        {
            k=k+i;
            ss[k]=0;
        }
        i=i+1;
        while(ss[i]==0)
            i=i+1;
    }
    for(i=1; i<=20; i++)
        if(ss[i]!=0)
            cout<<ss[i]<<" ";
    cout<<endl;
    return 0;
}
```

输出: 2 4 6 8 10 12 14 16 18 20

```
#include <iostream>
using namespace std;
int main()
{
    long i, j, a;
    long s[33];
    cin >> a;
    j = 0;
    while (a != 0)
    {
        j++;
        s[j] = a % 2;
        a = a / 2;
    }
    if (j == 0)
        cout << 0;
    else
    {
        for (i = j; i >= 1; i--)
            cout << s[i];
    }
    return 0;
}
```

输入：
58
输出： 111010

习题

1.有以下程序：

```
#include<stdio.h>
main( )
{    char str[ ]="tudent";   int i;
     for( i=1; str[i]!='\0'; i++)
```

```
  {    switch( str[i] )
       {   case 't':putchar('#');
           case 'n':putchar('$');  break;
           default :continue;
       }
       putchar('*');
   }
}
```

程序运行后输出的结果是($*#$* A)。

A) $*#$* B)$#$ C)#$*$*#$* D)#$$#$

2. 有以下程序：

```
main( )
{  char  x[6]={'a', 'c', 'c', 'e', 'a', 'c'}; int i=0;
   while( i<6 )
   {   if( x[i]+2=='c') printf("%4d",i);
       i++;
   }
}
```

程序运行后的输出结果是(A)。

A)0 4 B)1 2 5 C)1 5 D)2 3 4 5

3.以下程序的运行结果是(B)。

```
main( )
{ char s[6]= "12a34";
  int i,t=0;
  for(i=0;s[i]>= '0' && s[i]<= '9';i++) t=t*10+s[i]-'0';
  printf("t=%d\n",t);
}
```

A)1234 B)12 C)34 D)10

4.桶排序算法的时间复杂度 T(M, N)是多少？ ( )

```
void Bucket_Sort(ElementType A[], int N)
{   count[]初始化;
    while (读入 1 个学生成绩 grade)
        将该生插入 count[grade]链表;
    for ( i=0; i<M; i++ ) {
        if ( count[i] )
            输出整个 count[i]链表;
    }
}
```

A. O(M) B.O(N) C.O(MN) D. O(M+N)

5 桶排序的思想是若待排序的记录的关键字在一个明显有限范围内(整型)时，设计有限个有

序桶，每个桶装入一个值(也可装入若干个值)，顺序输出各桶的值，将得到有序的序列。

```
#include<iostream>
#include <cstring>
using namespace std;
int main(){
int b[101],k,i,n;
______1___________
cin>>n;
for( i=1;i<=n;i++)      {
cin>>k;
______2_________  //将关键字等于 k 的值全部装入第 k 桶
 }
 for( i=0; i<=100;i++)
    while (____3__________)  {
        cout<<i<<" " ;
        ________4___________;//
    }    //输出排序结果
    cout<<endl;
}
```

6.

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
    int a[9];
    int i, j, n;
    n = 8;
    for (i = 1; i <= n; i++)
        cin >> a[i];

    i = 0;
    for (j = 1; j <= n; j++)
    {
        if (a[j] % 3 == 0)
        {
            i = i + 1;
            a[i] = a[j];
        }

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

```
        cout << a[i] << " ";
    cout << endl;
    return 0;
}
```

输入：8 151 9 7 233 68 514 12

输出:_____9 12_____

7.

```
#include <iostream>
using namespace std;
int main()
{
    int n, m, i, j;
    char a[51][51];
    int  b[6]= {0};
    cin>>n>>m;
    n= n*5+1;
    m=m*5+1;
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=m; j++)
            cin>>a[i][j];
    }
    for(i=1; i<=n; i++)
        for(j=1; j<=m; j++)
            if ((a[i][j]!= '#')&& (i%5==2) && (j %5==2))
            {
                if(a[i][j]== '.')b[1]=b[1]+1;
                else if ((a[i][j]== '*')&& (a[i+1][j]== '.'))b[2]++;
                else if ((a[i+1][j]== '*')&& (a[i+2][j]== '.')) b[3]++;
                else if ((a[i+2][j]== '*')&& (a[i+3][j]== '.')) b[4]++;
                else if ((a[i+3][j]== '*')) b[5]++;
            }
    for(i=1; i<=5; i++)cout<<b[i]<<" ";
    return 0;
}
```

输入 1:

```
1  2
############
#....#*****#
#....#*****#
#....#*****#
#....#*****#
```

```
######################
#**************#....#
#**********#....#....#
#....#....#....#....#
############
```

输出 1：　　1 0 0 0 1

输入 2：

2 4

```
######################
#*****#*****#*****#*****#
#*****#....#*****#*****#
#....#....#*****#*****#
#....#....#....#*****#

#....#....#....#....#
######################
```

输出 2：

第十四讲　模拟

现实中有些问题难以找到公式或规律来求解，只能按照一定的步骤不停的"模拟"下去，最后才能得到答案。这样的问题，用计算机来求解十分合适，只要能让计算机模拟人在解决此问题时的行为即可。这种求解问题的思想，可以称为"模拟"。

1 [7335] 采花生

在参加“采花生” 这个项目比赛时， 考官会出示一块 n 行、 m 列的花生田，上面一共种了 n*m 株花生苗。 每株花生植株下都结了一定数量的花生果， 比赛开始时选手站在第 1 行， 第 1 列的位置， 现要求用**最短的时间找到结花生果最多的一株花生植株（数据保证花生果最多的植株只有一株）** ， 然后按先向南（下）走， 再向东（右） 的路线顺序去采摘它的花生果， 沿路经过的其他花生植株下面的花生果也要一并采摘下来， 但不允许采摘没有路过的花生植株， 否则依犯规出局处理。 问这个选手一共可以采摘到多少粒花生果？

	第 1 列	第 2 列	第 3 列	第 4 列	第 5 列	第 6 列
第 1 行	5	7	4	5	1	13
第 2 行	9	6	3	2	8	7
第 3 行	10	14	0	1	9	4
第 4 行	4	6	9	18	25	0
第 5 行	3	1	2	9	0	2

如一块 n=5， m=6 的花生田，可以发现结花生果最多的那株花生植株在（4， 5）， 则选手采摘的顺序应为（1， 1） -（2， 1） -（3， 1） -（4， 1） -（4， 2） -（4， 3） -（4， 4） -（4， 5），一共采得的花生果粒数为 5+9+10+4+6+9+18+25=86。

输入第 1 行有两个整数 n 和 m(1 < n,m <= 100)， 表示花生田一共有 n 行 m 列。第 2 至 n+1 行， 每行有 m 个用空格隔开的整数， 第 i + 1 行的第 j 个整数 Pij（0 <= Pij <= 700） 表示花生田里植株(i, j)下花生的数目， 0 表示该植株下没有花生。

输出只有一行， 一个整数， 表示选手一共摘到的花生果数目。

样例输入

5 6
5 7 4 5 1 13
9 6 3 2 8 7
10 14 0 1 9 4
4 6 9 18 25 0
3 1 2 9 0 2

样例输出

86

思路： 首先遍历矩阵找出最大的位置记录横纵坐标，然后根据题目描述先向下走再向右走并且累加路径上的和即可。

```
#include<bits/stdc++.h>
```

```
using namespace std;
int n,m;
int a[105][105];
int sum;
int main()
{
    cin>>n>>m;
    int maxn = 0,k1,k2;
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= m; j++){
            cin>>a[i][j];
            if(maxn<a[i][j]){
                k1 = i;//记录横纵坐标
                k2 = j;
                maxn = a[i][j];
            }//找到最大位置
        }
    }

    for(int i = 1; i <= k1; i++){
        sum+=a[i][1];
    }//向下走采摘的总值
    for(int i = 2; i <= k2; i++){
        sum+=a[k1][i];
    }//向右走采摘的总值（去掉第一个已经算过的）

    cout<<sum;
}
```

2 [1328] 校园健身走

2008 年是中国奥运年。为唱响“全民健身与奥运同行”这一主题，浙江财经大学于 4 月 1 日在下沙钱塘江堤举办了全体教职工“江堤健身走”活动。

活动规则如下：

1. 健身走的路线是从下沙校区南大门(起点)→校区东边钱塘江江堤→返回南大门(终点)，形成一个环线。在这一段环线上已经设置好 N 个景点(含起点)。

2. 每名参加“健身走”活动的教职工首先在起点领取一张活动券，并加盖起点的标识章。然后沿着路线步行，每到达一个景点，都需要加盖该景点的标识章，到达终点后，领取本次活动的纪念品。

你的任务是编写程序，计算本次“健身走”活动所需的时间。用到的信息如下：

1. 起点和终点都是南大门，包含起点在内，一共 N 个景点。这 N 个景点构成一个环线，一共 N 段路程，这些路程是已知的。

2. 整个活动开始时刻设为第 0 分钟。

3. 一共 M 批老师参加活动，假设每批老师步行速度是一致的，同时到达起点参加活动，同时抵达终点。每批老师的到达起点的时刻(分钟)、步行速度(单位距离/分钟)是已知的。

4. 每批老师步行整个路程所花的时间中，不足一分钟的以一分钟记。另外，每批老师在起点处领取活动券并加盖起点的标识章需 1 分钟，在其他景点加盖该景点的标识章需 1 分钟，到达终点后领取纪念品的时间也是 1 分钟。注意，如果存在若干批老师同时抵达一个景点，领取活动券、加盖标识章、领取纪念品等，所需的时间还是一分钟。

输入每个测试数据的第 1 行为一个正整数 N，表示本次活动景点的个数，N 不超过 100。
第 2 行为 N 个正整数，表示 N 段路程的距离，这些正整数的值都不超过 100。
第 3 行为一个正整数 M，表示参加活动的老师批数，M 不超过 50。
接下来的 M 行，每行为两个正整数 s 和 v，分别表示这批老师到达起点的时刻，及步行速度，s 的值不超过 100，v 的值不超过 10。

输出对每个测试数据，输出本次活动所需的时间(分钟)。

样例输入

```
6
20 30 40 50 60 70
3
0 5
9 4
6 10
```

样例输出

```
84
```

思路：总时间 t 由 3 部分 t1,t2,t3 组成，t1 为出发时间，t2 为走路时间，t3 为盖章时间。t1 已知，直接输入；t2=sum/v，即总路程除以速度，注意不到一分钟要算一分钟；t3=n+1，即总盖章时间。最后求出最大的 t。

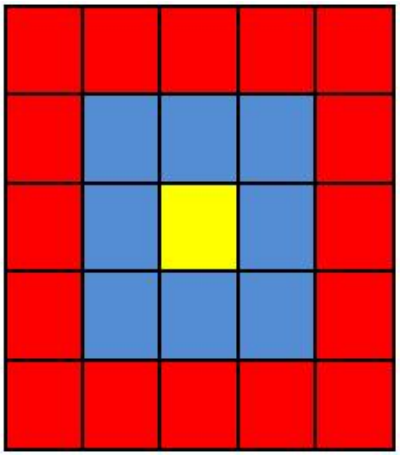
```
#include<iostream>
using namespace std;
int main(){
        int sum=0,n,m,t,t1,t2,t3,x,maxt=0;
        int i,v;
        cin>>n;//n 个景点
        for(i=1;i<=n;i++){//输入 n 个景点的距离，并求和
                cin>>x;
                sum=sum+x;
        }
        cin>>m;//m 批同学
        for(i=1;i<=m;i++){//输入 m 批同学
                cin>>t1>>v;//输入每批同学的出发时间和速度
                t2=sum/v;//求 t2，走路的时间
                if(sum%v!=0)t2++;//走路的时间不足一分钟算 1 分钟
```

```
            t3=n+1;//盖章的时间
            t=t1+t2+t3;//总时间
            if(t>maxt)maxt=t;//求每批的最大时间
        }
        cout<<maxt;
        return 0;
}
```

2 [7321] 花坛

洛洛在散步的时候，看到公园的正方形花坛里开放着许多他不认识的花卉。仔细观察之后，他又发现这些花的种植位置是有规律的。

洛洛发现在正方形花坛的最外层，即第一层上的花都是同一颜色；而花坛的第二层，花的颜色又都是一样的……正方形花坛由若干层花构成，同一层上的花都是同一颜色的，不同层之间的花颜色不一定相同。如下图所示，是一个具有三层花的正方形花坛：



在回到家后，洛洛还记得花坛有几层花围成，以及每层花的颜色，花的颜色用英文大小写字母来表示。但是洛洛忘记了整个花坛的图像，洛洛希望你根据他的描述，把整个花坛的图像用计算机打印字符的方式表示出来。

输入

第一行输入一个整数 n，表示正方形花坛有 n 层花。第二行输入 n 个字符，第 i 个字符表示第 i 层花的颜色。第一层是花坛最外层。第 n 层是花坛最内层，只有一朵花。

输出

输出 2*n-1 行，由(2*n-1)*(2*n-1)个字符组成的花坛的图像。

样例输入

3

abC

样例输出

aaaaa

abbba

abCba

abbba

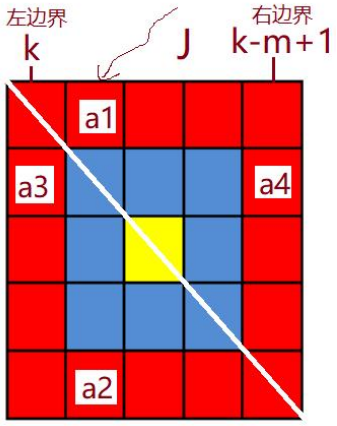
aaaaa

提示

第一层是颜色为字符 a 的花，在最外层；第二层是颜色为字符 b 的花，在第二层。第三层是颜色为字符 c 的花，只有一朵，在最内层。

思路: 这题的难点在于如何清晰的表达题目描述的规律。

1. 先输入最外层的颜色。先确定左边界和右边界，初始最外层的颜色的左边界为 k=1，右边界 2*n-1(或者总行数 m=2n-1)，接着循环依次赋值。理论上我们需要四个循环实现四条边的赋值，但是这是一个正方形，存在对称关系。如图所示， 赋值 a1 后，根据对称性，可生成 a2,a3,a4。



2. 确定对称位置的坐标，如果 当前 a1 的坐标为 a[k][j]。则

a1: a[k][j];//第 k 行第 j 列

a2= _____

a3= _____

a4= _____

3. 进入下一轮颜色赋值。此时变化的状态是，左边界需要加一即 k=k+1，右边界需要减一,，总行数 m=m-2。

```
#include <bits/stdc++.h>
using namespace std;
char a[2*1000][2*1000];
int main()
{
    int n,m,k=1;
    char x;
    cin>>n;
    m=2*n-1;//当前颜色的总行数
    for(int i=1;i<=n;i++)
    {
        cin>>x;//花的颜色
        for(int j=k;j<=k+m-1;j++)
        {////将花坛的每一格颜色记录
            a[k][j]=x;
            a[n*2-k][j]=x;
            a[j][k]=x;
            a[j][n*2-k]=x;
        }
        m-=2;//下一个颜色少两个
        k++;
    }
    for(int i=1;i<2*n;i++)
    {
        for(int j=1;j<2*n;j++)
            printf("%c",a[i][j]);
        printf("\n");
    }
    return 0;
}
```

```
a1  a[k][j]=x;
a2  a[n*2-k][j]=x;
a3  a[j][k]=x;
a4  a[j][n*2-k]=x;
```

写法二：

```
#include <iostream>
using namespace std;
char map[2001][2001];
int main()
{
	int n,c=0,x=1;
	string str;
	cin>>str;
	for(int i=1;i<=n;++i)
	{
		char c=str[i-1];
		for(int j=i;j<=n+n-i;++j)
			map[i][j]=c;
		for(int j=i;j<=n+n-i;++j)
			map[n+n-i][j]=c;
		for(int j=i+1;j<=n+(n-i-1);++j)
			map[j][i]=c;
		for(int j=i+1;j<=n+(n-i-1);++j)
			map[j][n+n-i]=c;
	}
	for(int i=1;i<=n*2-1;++i)
	{
		for(int j=1;j<=n*2-1;++j)
			cout<<map[i][j];
		cout<<endl;
	}
	return 0;
}
```

4 [7341] 懒羊羊吃草

总所周知，懒羊羊是所有小羊里最贪吃的一只。然而，鲜为人知的是，懒羊羊也有存储粮食的习惯。而更让大家吃惊的事实是，我们懒羊羊做事很有条理，每当他存储一份粮食时，他会专门拿出一个筐来存放。因此，他的仓库有很多很多筐的青草。而我们的懒羊羊又是一个经常馋嘴的小羊，**每当他想吃草时，就会从仓库里找出数量最少的一筐草，把它吃掉**。可是懒羊羊因为草吃的太多了导致大脑运转缓慢，所以他不得不向你请求支援，帮他找出他应该吃数量为多少的青草。

输入：

第一行为一个正整数 n，表示懒羊羊一共进行了 n 次操作（2<=n<=10000）。第二行到第 n+1 行每行表示一个懒羊羊的操作，当这行形式为单独一个字符**‘q’时，表示懒羊羊肚子饿了，要吃掉仓库中当前数量最少的那份青草**；当这行形式为一个字符**‘i’和一个正整数 k 时，表示懒羊羊将一份数量为 k（1<=k<=maxlongint）的青草存入了仓库**，‘i’和‘k’之间用

空格隔开。输入的数据保证每次询问时仓库里都有草可吃且所有的操作中懒羊羊至少会吃一次草。

输出： 每当输入‘q’时，输出懒样羊当前吃掉的那份青草的数量是多少。
样例输入
5
i 5
i 2
q
i 9
q
样例输出
2
5

分析：
当输入 q 时，经过排序后，懒羊羊吃掉最少的青草，输出并将该筐青草从数组除去。当输入 i 时，在数组的末尾加入一个重量为 w 的青草。

代码

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n,f[10005];
    int front=1,last=0;
    scanf("%d",&n);
    while(n--)
    {
        char ch;
        cin>>ch;
        if(ch=='i')
        {
            int w;
            cin>>w;
            f[++last]=w;//将新的青草加入数组
        }
        else//吃草
        {//排序，小的在前
            sort(f+front,f+last+1);
            printf("%d\n",f[front]);
            front++;//将吃了青草除去
        }
    }
```

```
    return 0;
}
```

5 [3352] 猴子选大王

N 只猴子选大王。选举办法如下：从头到尾 1、2、3 报数，凡报 3 的退出，余下的从尾到头 1、2、3 报数，凡报 3 退出；余下的又从头到尾报数，还是报 3 的退出；依此类推，当剩下的两只猴子时，取这时报数报 1 的为王。若想当猴王，请问当初应占据什么位置？

输入

猴子总数 N，N<1000。

输出

猴王所在的位置。

样例输入

10

样例输出

8

提示：十只猴子编号：1 2 3 4 5 6 7 8 9 100 编号，则出圈的次序为 3 6 9 7 2 5 4 10　剩下 8 和 1 时，8 号猴子报 1 为大王。

分析：按题意要去，先从左到右三的倍数就出列；再从右到做左，把三的倍数出列，最后剩下 2 个。

1. 如何表示 3 的倍数出列？_____

2. 如何实现从右到做的遍历？_____

3. 出列后如何实现重现编号？_____

```
#include <iostream>
using namespace std;
int a[1100];
int main()
{
        int n;
        cin>>n;
        for(int i=1;i<=n;i++)a[i]=i;//初始化
        while(n>=3)
        {//如果数组元素的个数>3，就继续处理
                for(int i=1;i<=n;i++)
                        if(i%3==0) a[i]=0;//从左到右扫描，碰到 3，改为 0
                for(int i=1;i<=n;i++)
                        if(a[i]==0)
                        {//更新数组，删除被 3 整除的元素
                                for(int j=i;j<n;j++)
                                a[j]=a[j+1];
                                n--;//更新数组长度
                        }
```

```
            for(int i=1;i<=n/2;i++)
            {//交换数组前后元素。第二轮为从后往前。
                int b=i-1;
                int temp;
                temp=a[i],a[i]=a[n-b],a[n-b]=temp;
            }
        }
        cout<<a[1]<<endl;//输出剩下的第一个元素
        return 0;
}
```

课前练习

1.

```
#include <iostream>
#include <string>
using namespace std;
int main(){
    int n=2,m=22,i,t,k,sum;
    sum=0;
    for(i=n;i<=m;i++){
        t=i;
        while(t>0){
            if(t%10==2)
                sum=sum+1;
            t=t/10;
        }
    }
    printf("%d",sum);
    return 0;
}
```

输出_____

2.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string a, t;
    int i, j;
    a = "NOIP2013";
    j = 0;
    for (i = 1; i <= 7; i++)
        if (a[j] > a[i])
            j = i;
    j = j - 2;
    for (i = 0; i <= j; i++)
        cout << a[i];
    return 0;
}
```

输出：_____

习题

1.读程序写结果_____

```
#include <iostream>
using namespace std;
 int main()
 {
     double b[6]={1.1,2.2,3.3,4.4,5.5,6.6},t;
```

```
        int i;
        t=b[0];
        for(i=0;i<5;i++)
            b[i]=b[i+1];
        b[5]=t;
        for(i=0;i<6;i++)
             printf(“%6.2f”,b[i]);
    }
```

2. 有以下程序，则程序的输出结果是_____

```
#include <iostream>
using namespace std;
 int main()
 {    int p[7]={11,13,14,15,16,17,18},i=0,k=0;
      while(i<7 && p[i]%2)
      { k=k+p[i]; i++;}
      printf("k=%d\n",k);
 }
```

3 [NOIP2001] 输入 n 个 0 到 100 之间的整数，由小到大排序输出，每个数占 4 格，每行输出占 8 个。

```
#include <iostream>
#include <cstdio>
using namespace std;
int I, j, k, n, x, b[101];
int main(){
    scanf(“%d\n”, &n);
    for(i=0; i<=100; i++) b[i]=0;
    for(i=1; i<=n; i++){
        scanf(“%d”, &x);
        b[x]=_____①_____;
    }
    ____②____;
    for(i=0; i<=100; i++)
        while(____③_____){
            printf(_____④_____);
            k=k+1;
            b[i]=b[i]-1;
            if(_____⑤_____) putchar( ‘\n’ );
        }
    putchar( ‘\n’ );
```

```
    return 0;
}
```

4.

```
#include <iostream>
using namespace std;
int main()
{
    int i, j, k, x, n, ans, s;
    int a[501];
    bool f;
    cin >> n >> x;
    for (i = 1; i <= n; i++)
        cin >> a[i];
    i = 1,j = n,f = false;
    s = 0;
    while (i <= j && !f)
    {
        k = (i + j) / 2;
        s = s + 1;
        if (x == a[k])
        {
            ans = k;
            f = true;
        }
        else if (x < a[k])
            j = k - 1;
        else
            i = k + 1;
    }
    if (f)
        cout << ans << " " << s;
    else
        cout << s;
    return 0;
}
```

输入 1：

```
4 11
7 9 11 17
```

输出 1：_____

输入 2：

```
20 256
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 101 105 109 110 138
```

输出 2：_____

第十五讲 综合应用

1[3327] 小鱼比可爱

人比人，气死人；鱼比鱼，难死鱼。小鱼最近参加了一个“比可爱”比赛，比的是每只鱼的可爱程度。参赛的鱼被从左到右排成一排，头都朝向左边，然后每只鱼会得到一个整数数值，表示这只鱼的可爱程度，很显然整数越大，表示这只鱼越可爱，而且任意两只鱼的可爱程度可能一样。由于所有的鱼头都朝向左边，所以每只鱼只能看见在它左边的鱼的可爱程度，它们心里都在计算，**在自己的眼力范围内有多少只鱼不如自己可爱**呢。请你帮这些可爱但是鱼脑不够用的小鱼们计算一下。

输入
第一行输入一个整数 n，n<=100,表示鱼的数目。
第二行内输入 n 个整数，用空格间隔，依次表示从左到右每只小鱼的可爱程度。
输出
行内输出 n 个整数，用空格间隔，依次表示每只小鱼眼中有多少只鱼不如自己可爱。
样例输入
6
4 3 0 5 1 2
样例输出
0 0 0 3 1 2

思路：如果当前数为 a[i],统计左边[1~i-1]小于 a[i]的数的个数。

```
#include<iostream>
using namespace std;
int a[101],b[101],n;
int main()
{
        cin>>n;
        for (int i=1;i<=n;i++)//读入每条鱼的可爱值
                cin>>a[i];
        for (int i=1;i<=n;i++)//枚举 N 条鱼
                for (int j=i;j>=1;j--)//从第 I 个位置倒着往前找
                {
                        if (a[j]<a[i])
                        b[i]++;//如果找到比第 I 条鱼丑的，统计数组 b 对应的 b[i]+1
                }
        for (int i=1;i<=n;i++) cout<<b[i]<<" ";//输出
        return 0;
}
```

2 [3330] 统计天数

炎热的夏日，KC 非常的不爽。他宁可忍受北极的寒冷，也不愿忍受厦门的夏天。最近，他开始研究天气的变化。他希望用研究的结果预测未来的天气。
经历千辛万苦，他收集了连续 N（1<=N<=10^7）天的最高气温数据。
现在，他想知道最高气温一直上升的最长连续天数。
输入
1 行：一个整数 N。1<=N<=10^7
2 行：N 个空格隔开的整数，表示连续 N 天的最高气温。0<=最高气温<=10^9。
输出
1 行：一个整数，表示最高气温一直上升的最长连续天数。
样例输入
10
1 2 3 2 4 5 6 8 5 9
样例输出 5

思路：求气温一直上升的最长连续天数，也就是求序列最大连续上升序列长度。初始 mmax=0,ans=1,分别表示最大长度值，当前长度。从当前元素 a[i]开始算，此时一个元素 a[i]的长度为 1。如果 a[i]<a[i+1],即后面的元素比它大，长度 ans 加 1，如果不满足达到则求局部最优值 mmax,同时 ans=1，从新计算下一个 ans,一直处理到最后一个元素为止。做此类题目需要考虑边界，比如最大长度处于末端的时候。

```
#include<bits/stdc++.h>//头文件
using namespace std;//开辟名称空间
int N,a[10000005],ans=1,mmax=0;//注意 ans 要从 1 开始
int main()//开始
{
        scanf("%d",&N);//输入
        for( int i=0;i<N;i++)
        {
                scanf("%d",&a[i]);//输入
        }
        for( int j=0;j<N;j++)
        {
                if(a[j]>=a[j+1])//逐个判断要是开始升了就开始终止这一段
                {
                        if(ans>mmax)mmax=ans;//判断是否大于最大值
                        ans=1;//注意从 1
                }
                if(a[j]<a[j+1])ans++;
        }
        printf("%d",mmax);//输出
```

```
        return 0;//结束
}
```

3 [1331] 删除字符串中指定字符

题目描述
输入两个字符串 s1 和 s2，在 s1 中删除任何 s2 中有的字符。例如，s1 为"abc123ad"，s2 为"a1"，则输出"bc23d"。
输入
输入文件中包含多个测试数据。输入文件中第 1 行为一个正整数 T，表示输入文件中测试数据的数目。每个测试数据占一行，为两个字符串 s1 和 s2（s1,s2 的长度都<=100)，用空格隔开。

注意：本题的字符串由大写字母 A~Z,小写字母 a~z,以及数字 0-9 组成，对于同一字母的大小写（如：a 和 A）认为是不同的字符
输出
对输入文件中的每个测试数据，输出删除后的字符串 s1。
样例输入
3
abc123ad a1
aaaaaaaaaa a
baaaaaaaabb a
样例输出
bc23d

Bbb

思路：两重循环，在字符串 str1 中查找： 在 str2 中出现的每个字符 str2[i]。如果出现就做标记，全部做标记后重新扫描 str1,把没标记的字符输出即可。

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
        char s1[100],s2[100];
        int i,j,t,k;
        cin>>t;
        for(k=0;k<t;k++){
                cin>>s1>>s2;
                  for(i=0; i<strlen(s1); i++)
                        for(j=0; j<strlen(s2); j++)
                        {                              //把 s1 中，所有等于
                                if(s1[i]==s2[j])//s2[j]的字符 s1[i]改成空格
                                s1[i]=' ';
```

```
                }

            for(i=0; i<strlen(s1); i++)
            if(s1[i]!=' '){//不输出空格，表示删除
                cout<<s1[i];
            }
            cout<<endl;
        }
        return 0;
}
```

4 [2708] 连续出现的字符

题目描述
给定一连续出现的字符个字符串，在字符串中找到第一个连续出现至少 k 次的字符。
输入
第一行包含一个正整数 k，表示至少需要连续出现的次数。1 ≤ k ≤ 1000。
第二行包含需要查找的字符串。字符串长度在 1 到 2500 之间，且不包含任何空白符。
输出　若存在连续出现至少 k 次的字符，输出该字符；否则输出 No。

样例输入
3
abcccaaab
样例输出
c

思路：第一个连续出现至少 k 次的字符，这里的重点是连续，所以要统计当前字符连续出现的次数。

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char str[80];
    int k,i,s=1,flag=1;
    cin>>k>>str;
    for(_1__________________){//扫描每一个字符
        if(_2_______________)//如果相等，跟前一个还是后一个？
            s++;
        if(s>=k){//如果次数>=k
            cout<<str[i]<<endl;
            _3_____________
            break;
        }
```

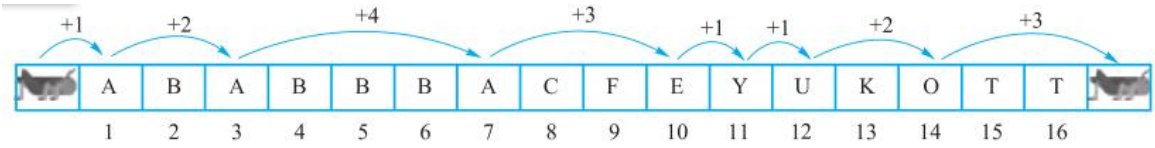
```
        if(_4____________________)//如果出现不相等，归 1
            s=1;
    }
    if(flag==1)
        cout<<"No"<<endl;
    return 0;
}
```

5［6865］蚱蜢

有一天，一只蚱蜢像往常一样在草地上愉快地跳跃，它发现了一条写满了英文字母的纸带。

蚱蜢只能在元音字母(A、E、I、O、U、Y)间跳跃，一次跳跃所需的能力是两个位置的差。纸带所需的能力值为蚱蜢从纸带开头的前一个位置根据规则跳到纸带结尾的后一个位置的过程中能力的最大值。

蚱蜢想知道跳跃纸带所需的能力值(最小)是多少。如图 9.3-1 所示的纸带所需的能力值(最小)是 4。



[输入格式]

一行一个字符串，字符串长不超过 100。

[输出格式]

一行一个整数，代表(最小)能力值。

[输入样例]

KMLPTGFHNBVCDRFGHNMBVXWSQFDCVBNHTJKLPMNFVCKMLPTGFHNBVCDRFGHNMBVXWSQFDCVBNHTJKLPMNFVC

[输出样例]

85

分析：从头到尾枚举纸带的每一个字母，按照规则模拟蚱蜢在元音字母之间跳跃的过程，打擂台记录能力值。

```
 #include<bits/stdc++.h>
using namespace std;
char t[101];
int main()
{
    scanf("%s",t+1);
    int n=strlen(t+1);
    int ans=0,x=0;
    for(int i=1;i<=n;i++)
```

```
    {
        if(t[i]=='A'||t[i]=='E'||t[i]=='O'||t[i]=='U'||t[i]=='I'||t[i]=='Y')
        {
            ans=max(ans,i-x);
            x=i;
        }
    }
    cout<<ans<<endl;
}
```

6［7319］听歌识曲

洛洛有一份私人歌单，歌单里面塞满了他喜欢的歌曲，像夏恋、雨道、彩月、幻昼……整整有好几百首。洛洛每天都要把他的歌单听一遍，以致于他都能知道在什么时候放的是什么歌。洛洛在向你推荐了他的歌单之后，决定考考你，**从他的歌单开始播放起，第 t 秒正在播放的是第几首歌。**

输入

第一行输入两个整数 n 和 t， 分别表示歌单的歌曲总数以及第 t 秒播放哪首歌。

第二行有 n 个整数， A1， A2， …， An， 分别表示歌单的第 i 首歌将会播放多长时间。

1 ≤ n ≤ 100000， 1 ≤ Ai ≤ 1000， 1 ≤ t ≤ sum(Ai)

输出

输出一个整数，表示歌单按顺序播放后，第 t 秒播放的是第几首歌。样例输入

3 5

1 3 5

样例输出

3

提示

歌单中总共有三首歌：

第一首歌播放 1 秒，占第 1 秒；第二首歌播放 3 秒，占第 2-4 秒；第三首歌播放 5 秒，占第 5-9 秒。所以第 5 秒播放的是第三首歌曲。

思路：从前往后的累加，就可以知道每首歌结束的时候在第几秒，用一个数组来记录每首歌结束的时间（第 i 首歌结束的时间为 a[i]秒）。最后找到第一个结束时间晚于指定时间 t 的歌，输出它的下标即可。

```
#include<bits/stdc++.h>
using namespace std;
int a[100005];
int main()
{
    int n,t,m,i,j,sum=0;a[0]=0;
    scanf("%d %d",&n,&t);//一共有 n 首歌，求第 t 秒播放的歌
    for(i=1;i<=n;i++){
```

```
        scanf("%d",&m);
        sum=sum+m;//每次累加，即为每首歌结束的时间
        a[i]=sum;//将每次累加的值赋值给数组 a[]
    }
    for(i=1;i<=n;i++){//从头开始找
        if(a[i-1]<t&&a[i]>=t){
            printf("%d",i);break;//找到第一首结束的时间晚于指定时间的歌,输出
下标
        }
    }
    return 0;
}
```

7 [7320] 多面骰子

洛洛现在手上有三颗多面骰子，多面骰子不是常见的六面骰子，而是 33 面骰子、100 面骰子……一般来说， i 面骰子每个面上的点数分别是 1， 2， 3，……i。
洛洛手上的三颗骰子的面数可能并不相同，他想知道掷出三颗骰子的所有情况中，三颗骰子的点数之和出现最多次数是几点。
如果存在多个点数之和出现次数相同的情况，则按点数之和从小到大顺序输出。
输入
第一行输入三个整数 n1， n2， n3，分别表示三颗骰子各自的面数。1 ≤ n1， n2， n3 ≤ 100。

输出
输出一行含任意个整数，分别表示次数最多的点数之和，用空格隔开。
样例输入
1 2 3
样例输出
4 5
提示
样例 1，骰子投出来有以下六种情况：
1+1+1=3 1+1+2=4 1+1+3=5
1+2+1=4 1+2+2=5 1+2+3=6
可以看出 4 和 5 的出现次数最多且都是 2 次。
题目类型：枚举
思路：
因为这道题目的数据范围很小，我们可以直接枚举出所有的情况是可以的。那就直接用三层循环，枚举出所有可能的点数和，用一个数组来记录每种结果出现的次数。最后输出出现次数最多的点数。

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

```
int a[1000];
int main(){
    int n1,n2,n3,i,j,k,sum=0,max=0;
    scanf("%d %d %d",&n1,&n2,&n3);//输入三个骰子的面数
    for(i=1;i<=n1;i++){
        for(j=1;j<=n2;j++){
            for(k=1;k<=n3;k++){
                sum=i+j+k;//sum 即为对应的点数和
                a[sum]++;//对应的点数和出现次数加一
                if(a[sum]>max)max=a[sum];//用 max 来记录出现的最多次数
            }
        }
    }
    for(i=1;i<305;i++){//题目给出最多有 100 面，那么点数最大为 300
        if(a[i]==max)printf("%d ",i);//输出出现次数为 max 的所有点数
    }
    return 0;
}
```

课前练习

1.

```
#include <iostream>
using namespace std;
int main()
{
    int i, j, k, t;
    int a[8];
    for (i = 1; i <= 7; i++)
        a[i] = 0;
    for (i = 1; i <= 4; i++)
        a[i] = i;
    t = a[7];
    for (i = 7; i >= 2; i--)
        a[i] = a[i - 1];
    a[1] = t;
    for (i = 1; i <= 7; i++)
        cout << a[i];
    return 0;
}
```

输出：_____0123400_____

2.

```
#include <iostream>
using namespace std;
int main(){
        char str[1000];
        int len;
        int i;
        gets(str);
        len=strlen(str);
        for(i=0;i<len;i++){
                if(str[i]>='a'&&str[i]<='z')
                        str[i]-=32;
        }
        puts(str);
        return 0;
}
```

输入：helloworld123Ha

输出：_____

HELLOWORLD123HA

习题

1.

```
int main()
{
        char ch[100];
        gets(ch);
        int num=0,i;
        for(i=0;i<strlen(ch);i++)
                if(('A'<=ch[i]&&ch[i]<='Z')||('a'<=ch[i]&&ch[i]<='z'))
                        num++;
        printf("%d",num);
        return 0;
}
```

输入： A C B B Babcs

输出_____**9**_____

2.炎热的夏日，KC 非常的不爽。他宁可忍受北极的寒冷，也不愿忍受厦门的夏天。最近，开始研究天气的变化。他希望用研究的结果预测未来的天气。经历千辛万苦，他收集了连续 N（1<=N<=10^7）天的最高气温数据。现在，他想知道最高气温一直上升的最长连续天数。

输入*1 行：一个整数 N。1<=N<=10^7

*2 行：N 个空格隔开的整数，表示连续 N 天的最高气温。0<=最高气温<=10^9。输出*1 行：一个整数，表示最高气温一直上升的最长连续天数。

样例输入

10

1 2 3 2 4 5 6 8 5 9

样例输出

5

```
#include<bits/stdc++.h>//头文件
using namespace std;//开辟名称空间
int N,a[10000005],__1_________________
int main()//开始
{
        scanf("%d",&N);//输入
        for( int i=0;i<N;i++)
                scanf("%d",&a[i]);//输入
        for( int j=0;j<N;j++)
        {
                if(__2_____________)//逐个判断要是开始升了就开始终止这一段
                {
                        if(ans>mmax)__3_____________//判断是否大于最大值
                        ans=1;//注意从 1
```

```
                }
                if(a[j]<a[j+1])_4_____________________
        }
        printf("%d",mmax);//输出
        return 0;//结束
}
```

3 输出_____

```
#include<stdio.h>
int main(){
  int a=5;
  while(a!=1){
    if(a%2!=0){
        printf("%d*3+1=%d\n",a,a*3+1);
        a=a*3+1;
    }
    else{
        printf("%d/2=%d\n",a,a/2);
        a=a/2;
    }
  }
  printf("End\n");
  return 0;
}
```

```
5*3+1=16
16/2=8
8/2=4
4/2=2
2/2=1
End
```

4 阅读程序

```
#include <iostream>
#include <string>
using namespace std;
int i,j,len,s;
const int maxn=20;
int a[maxn+1], b[maxn+1];
string n;
int main()
{
    cin>>n;
    len=n.length();
    i=len-1;
    j=0;
    while(i>= 0)
    {
        a[j]=n[i]-'0';
        j++;
        i--;
```

```
    }
    s=0;
    for(i=0; i<len; i++)
    {
        if(a[i]>=5)
            s++;
    }
    cout<<s<<endl;
    return 0;
}
```

输入：

387543284570123

输出：6

5. getline(cin,s):接收一个字符串 s，且可以接收空格；

str1.compare(str2):比较两个字符串的大小，如果相等，输出 0，不等输出-1。

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string s,s1;
    int i,a,b,k;
    getline(cin,s);
    s1="you";
    k=0;
    a=s.length();
    b=s1.length();
    for(i=0; i<=a-b+1; i++)
    {
        if(s1.compare(s.substr(i,b))==0)
            k=k+1;
    }
    cout<<k<<endl;

    return 0;
}
```

输入：It’s easy. You get off the bus. Then you cross the road. You take the first road on the left. You walk for five minutes.

输出：_____1_____