第五章 数组

第一节 一维数组

第二节 二维数组

第三节 字符数组和字符串类型

第一节　一维数组

一、为什么要使用数组

通过前面几章的学习，我们已经可以编写程序来解决各种相当复杂的问题了，但是当需要处理的数据比较多时，仅依靠前面的知识是不够的，即使简单的问题也可能需要比较复杂的程序来处理。请看下面的例子：

例题：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】在解决这个问题时，虽然可以通过一个变量来累加读入的50个成绩求出学生的总分，进而求出平均分。但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a_1，a_2，…，a_{50}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>>a_1>>a_2>>…>>a_{10};

…

cin>>a_{41}>>a_{42}>>…>>a_{50};

注意，如果真正要像上面这样编写程序，则上面的所有省略号必须用完整的语句写出来。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。

选手们可能已经看出，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（在上面的例子中，各分量是整型变量或实型变量的性质）。

如果能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。如果使用数组，上面的问题就变得十分简单、清晰。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];
```

在这里引用了带下标的变量（分量变量称为数组元素）a[i]来代替a_1,a_2…,a_{50},方括号中的i称为下标，当循环变量i=1时a[i]就是a[1]；当i=2时a[i]就是a[2]……；当i=50时a[i]就是a[50]。输入的时候，让i从1变化到50，循环体内输入语句中的a[i]也就分别代表了a_1,a_2…,a_{50}这50个带下标的变量。这样上述问题的程序可写为：

```
tot = 0;                        // tot存储50个学生的总分
for (int i=1;i<=50;++i)  // 循环读入每一个学生的成绩，并把它累加到总分中
{
  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];
//如果第i个同学成绩小于平均分，则将输出这个学生的序号和成绩。
```

要在程序中使用下标变量，必须先说明这些下标变量的整体为数组，即数组是若干个同名（如上面的下标变量的名字都为a）下标变量的集合，这些变量的类型全部一致。

二、一维数组的定义

当数组中每个元素只带有一个下标时，我们称这样的数组为一维数组。

数组的定义格式如下：

类型标识符　数组名[常量表达式]

说明：

①数组名的命名规则与变量名的命名规则一致。

②常量表达式表示数组元素的个数。可以是常量和符号常量，但不能是变量。

例如：

```
int a[10];              //数组a定义是合法的
int b[n];               //数组b定义是非法的
```

三、一组数组的引用

通过给出的数组名称和这个元素在数组中的位置编号(即下标)，程序可以引用这个数组中的任何一个元素。

一维数组元素的引用格式：

数组名[下标]

例如：

```
int a[10];
```

其中，a是一维数组的数组名，该数组有10个元素，依次表示为：

a[0],a[1],a[2],a[3],a[4],a[5],a[6],a[7],a[8],a[9]。

需要注意的是：a[10]不属于该数组的空间范围。

当在说明部分定义了一个数组变量之后，C++编译程序为所定义的数组在内存空间开辟一串连续的存储单元。例如：
上例中的a数组在内存的存储如表所示：

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]

a数组共有10个元素组成，在内存中10个数组元素共占10个连续的存储单元。a数组最小下标为0，最大下标9。按定义a数组所有元素都是整型变量。

再次提醒注意：类型和变量是两个不同概念，不能混淆。就数组而言，程序的执行部分使用的不是数组类型而是数组变量。

说明：

(1)下标可以是整型常量或整型表达式。如果使用表达式作为下标，就要计算表达式的值以确定下标。

(2)C++语言中，每个数组第一个元素的下标都是0，因此第一个元素为第0个数组元素。

(3)C++语言只能逐个引用数组元素，而不能一次引用整个数组。

例如：int a[100],b[100]; a=b;这样的写法是非法的。

(4)数组元素可以像同类型的普通变量那样使用，对其进行赋值和运算的操作，和普通变量完全相同。

例如： c[10]=34;实现了给c[10]赋值为34。

四、一维数组的初始化

数组的初始化可以在定义时一并完成。格式：

类型标识符 数组名[常量表达式]={值1，值2，…}

例如：

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

说明：

(1)在初值列表中可以写出全部数组元素的值，也可以写出部分。例如，以下方式可以对数组进行初始化：

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

该方法一次仅对数组的前5个元素依次进行初始化。

(2)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。

五、一维数组的应用

例5.1　输入n个数,要求程序按输入时的逆序把这n个数打印出来，已知整数不超过100个。也就是说，按输入相反顺序打印这n个数。

【分析】我们可定义一个数组a用以存放输入的n个数，然后将数组a中的内容逆序输出。

```
#include<cstdio>
int a[100];
int main()
{
    int x,n=0;
    while(scanf("%d",&x)==1) a[n++]=x;  //相当{a[n]=x;n++;}
    for (int i=n-1;i>=1;--i)
        printf("%d ",a[i]);  //注意%d后面有一个空格，保证行首行尾均无空格
    printf("%d\n",a[0]);
    return 0;
}
```

【说明】：

语句int a[100]声明了一个包含100个整型变量的数组，它们是：a[0]，a[1]，a[2]，…，a[99]。注意，没有a[100]。在上述程序中，数组a被声明在main函数的外面。只有放在外面时，数组a才可以开得很大；放在main函数内时，数组稍大就会异常退出。它的道理将在后面讨论，只需要记住规则即可。

数组不能够进行赋值操作：如果声明的是int a[MAXN]，b[MAXN],是不能赋值b=a的(Pascal语言可以的)。如果要从数组a复制k个元素到数组b，可以这样做：memcpy(b,a,sizeof(int)*k)。当然了，如果数组a和b都是浮点型的，复制时要写成memcpy(b,a,sizeof(double)*k)。如果需要把数组a全部复制到数组b中，可以写得简单一些：memcpy(b,a,sizeof(a))。使用memcpy函数要包含头文件cstring。

例5.2 将a数组中第一个元素移到数组末尾,其余数据依次往前平移一个位置。

【分析】为完成题目所要求的操作，其算法应该包括以下几个主要步骤：

①把第一个元素的值取出放在一个临时单元 temp中；

②通过 a[2]→a[1], a[3]→a[2], a[4]→a[3]，……， a[n]→a[n-1],实现其余元素前移

③将 temp值送入a[n].

```
#include<iostream>
#include<iomanip>                                   //调用setw函数需注明使用该库
const int n=10;
using namespace std;
int a[n],temp;
int main()
{
    cout<<"read "<<n<<" datas"<<endl;
       for (int i=0; i<n; ++i) cin>>a[i];
       temp=a[0];
       for (int i=0; i<n-1; ++i) a[i]=a[i+1];
       a[n-1]=temp;
       cout<<"Result:"<<endl;
       for (int i=0; i<n; ++i) cout<<setw(3)<<a[i];   //setw函数控制输出场宽
       return 0;
 }
```

运行结果 :

read 10 datas:

- 1 2 3 4 5 6 7 8 9 10

Result:

- 2 3 4 5 6 7 8 9 10 1

例5.3 宾馆里有一百个房间，从1-100编了号。第一个服务员把所有的房间门都打开了，第二个服务员把所有编号是2的倍数的房间“相反处理”，第三个服务员把所有编号是3的倍数的房间作“相反处理”…，以后每个服务员都是如此。当第100个服务员来过后，哪几扇门是打开的。（所谓“相反处理”是：原来开着的门关上，原来关上的门打开。）

【分析】此题较简单，用a[1],a[2],…,a[n]表示编号为1，2，3，…,n的门是否开着。模拟这些操作即可，参考程序如下：

```
#include<cstdio>
#include<cstring>
#define MAXN 100+10
int a[MAXN];
int main()
{
        int n,k,first=1;
        memset(a,0,sizeof(a));
        for (int i=1;i<=100;++i)
          for (int j=1;j<=100;++j)
            if (j%i==0) a[j]=!a[j];
for (int i=1;i<=100;++i)
    if (a[i])
    {
        if(first) first=0;
            else printf(" ");
        printf("%d",i);
    }
printf("\n");
return 0;
}
```

运行结果：

1 4 9 16 25 36 49 64 81 100

【说明】：

memset(a,0,sizeof(a))的作用是把数组a清零，它在cstring中定义。虽然也能用for循环完成相同的任务，但是用memset又方便又快捷。另一个技巧在输出：为了避免输出多余空格，设置了一个标志变量first，可以表示当前要输出是否为第一个。第一个变量前不应该有空格，但其他都有。

例5.4 约瑟夫问题：N个人围成一圈，从第一个人开始报数，数到M的人出圈；再由下一个人开始报数，数到M的人出圈；…输出依次出圈的人的编号。N，M由键盘输入。

【分析】（1）由于对于每个人只有出圈和没有圈两种状态，因此可以用布尔型标志数组存储游戏过程中每个人的状态。不妨用true表示出圈，false 表示没有出圈。

（2）开始的时候，给标志数组赋初值为false，即全部在圈内。

（3）模拟报数游戏的过程，直到所有的人出圈为止。

程序如下：

```
#include<iostream>
using namespace std;
int n,m,s,f,t;
bool a[101];                          //根据题意开出数组大小
int main()
{
    cin>>n>>m;                        //共n人，报到m出圈
    cout<<endl;
    for (t=1;t<=n;++t) a[t]=false;    //等同于memset(a,0,sizeof(a)),要调用cstring库
    f=0;  t=0;  s=0;                  //刚开始所有变量默认值也是0,或者用f=t=s=0;
    do
    {
        ++t;                          //逐个枚举圈中的所有位置
        if (t==n+1) t=1;              //数组模拟环状，最后一个与第一个相连
        if (a[t]==false) ++s;         //第t个位置上有人则报数
        if (s==m)                     //当前报的数是m
        {
            s=0;                      //计数器清零
            cout<<t<<" ";             //输出出圈人的编号
            a[t]=true;                //此处的人已出圈，设置为空
            f++;                      //出圈的人数增加一个
        }
    } while(f!=n);                    //直到所有的人都出圈为止
  return 0;
}
```

运行结果：

输入： 8 5

输出： 5 2 8 7 1 4 6 3

这是一个在算法设计上很有名气的经典约瑟夫（Josephu）问题，它有很多变例。如猴子选大王、持密码报数、狐狸追兔子等（见上机练习）。

例5.5 输入十个正整数,把这十个数按由大到小的顺序排列。（选择排序 ）

将数据按一定顺序排列称为排序，排序的算法有很多，其中选择排序是一种较简单的方法。

【问题分析】

要把十个数按从大到小顺序排列，则排完后，第一个数最大，第二个数次大，……。因此，我们第一步可将第一个数与其后的各个数依次比较，若发现，比它大的，则与之交换，比较结束后，则第一个数已是最大的数。同理，第二步，将第二个数与其后各个数再依次比较，又可得出次大的数。如此方法进行比较，最后一次，将第九个数与第十个数比较，以决定次小的数。于是十个数的顺序排列结束。

如对5个进行排序，这个五个数分别为8 2 9 10 5。按选择排序方法，过程如下：

初始数据 ：8 2 9 10 5

第一次排序：8 2 9 10 5

9 2 8 10 5

10 2 8 9 5

10 2 8 9 5

第二次排序：10 8 2 9 5

10 9 2 8 5

10 9 2 8 5

第三次排序：10 9 8 2 5

10 9 8 2 5

第四次排序：10 9 8 5 2

对于十个数，则排序要进行9次。

程序如下：

```
#include<iostream>
#include<iomanip>
using namespace std;
int t,a[11];
int main()
{
   cout<<"Input 10 intergers:"<<endl;
                          //读入10个初始数据
    for (int i=1; i<=10; ++i) cin>>a[i];
    cout<<endl;
    for (int i=1; i<=9; ++i)              //进行9次排序
      for (int j=i+1; j<=10; ++j)
                          //将第i个数与其后所有数比较
         if (a[i]<a[j]) { t=a[i]; a[i]=a[j]; a[j]=t;}
                          //若有比a[i]大, 则与之交换
        for (int i=1;i<=10;++i)
            cout<<setw(5)<<a[i];
        return 0;
      }
```

运行结果：

输入： 8 67 52 189 74 5 58 9 23 41

输出： 189 74 67 58 52 41 23 9 8 5

例5.6　编程输入十个正整数，然后自动按从大到小的顺序输出。（冒泡排序）

【问题分析】

①用循环把十个数输入到A数组中；

②从A[1]到A[10]，相邻的两个数两两相比较，即：A[1]与A[2]比，A[2]与A[3]比，……A[9]与A[10]比。

只需知道两个数中的前面那元素的标号，就能进行与后一个序号元素（相邻数）比较，可写成通用形式A[i]与A[i+1]比较，那么，比较的次数又可用1～(n-i)循环进行控制(即循环次数与两两相比较时前面那个元素序号有关）；

③在每次的比较中，若较大的数在后面，就把前后两个对换，把较大的数调到前面，否则不需调换位置。

下面例举5个数来说明两两相比较和交换位置的具体情形：

5　6　4　3　7

5和6比较，交换位置，排成下行的顺序；

6　5　4　3　7

5和4比较，不交换，维持同样的顺序；

6　5　4　3　7

4和3比较，不交换，顺序不变

6　5　4　3　7

3和7比较，交换位置，排成下行的顺序；

6　5　4　7　3

经过（1～（n-1））次比较后，将3调到了末尾

经过第一轮的1～（N-1)次比较，就能把十个数中的最小数调到最末尾位置，第二轮比较1～（N-2)次进行同样处理，又把这一轮所比较的“最小数”调到所比较范围的“最末尾”位置；……；每进行一轮两两比较后，其下一轮的比较范围就减少一个。最后一轮仅有一次比较。在比较过程中，每次都有一个“最小数”往下“掉”，用这种方法排列顺序，常被称之为“冒泡法”排序。

程序如下：

```
#include<iostream>
#include<iomanip>
using namespace std;
const int n=10;
int t,a[n+1];                          //定义数组
int main()
{
  for (int i=1; i<=n; ++i) cin>>a[i];
                                       //输入十个数
   for (int j=1; j<=n-1; ++j)
                                       //冒泡法排序
   for (int i=1; i<=n-j; ++i)  //两两相比较
      if (a[i]<a[i+1])                 //比较与交换
        {t=a[i]; a[i]=a[i+1]; a[i+1]=t;}
   for (int i=1; i<=n; ++i)
        cout<<setw(5)<<a[i];
                              //输出排序后的十个数
   cout<<endl;
   return 0;
}
```

运行结果：

输入： 2 5 8 6 12 34 65 22 16 55

输出： 65 55 34 22 16 12 8 6 5 2

例5.7 用筛法求出100以内的全部素数，并按每行五个数显示。

【问题分析】

⑴ 把2到100的自然数放入a[2]到a[100]中（所放入的数与下标号相同）；

⑵ 在数组元素中，以下标为序，按顺序找到未曾找过的最小素数minp,和它的位置p（即下标号）；

⑶ 从p+1开始，把凡是能被minp整除的各元素值从a数组中划去（筛掉），也就是给该元素值置0；

⑷ 让p=p+1，重复执行第②、③步骤，直到minp>floor(sqrt(N)) 为止；

⑸ 打印输出a数组中留下来、未被筛掉的各元素值，并按每行五个数显示。

用筛法求素数的过程示意如下（图中用下划线作删去标志）：

① 2 3 4 5 6 7 8 9 10 11 12 13 14 15…98 99 100
//置数

② 2 3 4 5 6 7 8 9 10 11 12 13 14 15…98 99 100
//筛去被2整除的数

③ 2 3 4 5 6 7 8 9 10 11 12 13 14 15…98 99 100
//筛去被3整除的数

……

2 3 4 5 6 7 8 9 10 11 12 13 14 15…98 99 100
//筛去被整除的数

程序如下：

```
#include<iostream>
#include<math.h>
                //在Dev C++中可调用数学函数库cmath
#include<iomanip>
using namespace std;
const int n=100;
int t;
bool a[n+1];
int main()
{
  for (int i=0; i<=n; ++i) a[i]=true;
      //等同于memset(a,1,sizeof(a)) ,
        要调用cstrin库
    a[1]=false;
    for (int i=2; i<=sqrt(n); ++i)
      if (a[i])
        for (int j=2; j<=n/i; ++j)
          a[i*j]=false;
    t=0;
    for (int i=2; i<=n; ++i)
      if (a[i])
      {
          cout<<setw(5)<<i;
          t++;
          if (t%5==0) cout<<endl;
      }
     return 0;
}
```

【上机练习】

1、与指定数字相同的数的个数【1.6编程基础之一维数组01】

输出一个整数序列中与指定数字相同的数的个数。

输入:

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

【上机练习】

2.陶陶摘苹果【1.6编程基础之一维数组02】Noip2005普及组第1题

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。

输入:

包括两行数据。第一行包含10个100到200之间(包括100和200)的整数(以厘米为单位)分别表示10个苹果到地面的高度，两个相邻的整数之间用一个空格隔开。第二行只包括一个100到120之间(包含100和120)的整数(以厘米为单位)，表示陶陶把手伸直的时候能够达到的最大高度。

输出:

包括一行，这一行只包含一个整数，表示陶陶能够摘到的苹果的数目。

样例输入:

100 200 150 140 129 134 167 198 200 111

110

样例输出:

5

【上机练习】

3.计算书费【1.6编程基础之一维数组03】

下面是一个图书的单价表：

计算概论 28.9元/本	数据结构与算法 32.7元/本
数字逻辑 45.6元/本	C++程序设计教程 78元/本
人工智能 35 元/本	计算机体系结构 86.2元/本
编译原理 27.8元/本	操作系统 43元/本
计算机网络 56元/本	JAVA程序设计 65元/本

给定每种图书购买的数量，编程计算应付的总费用。

输入:

输入一行，包含10个整数(大于等于0，小于等于100)，分别表示购买的《计算概论》、《数据结构与算法》、《数字逻辑》、《C++程序设计教程》、《人工智能》、《计算机体系结构》、《编译原理》、《操作系统》、《计算机网络》、《JAVA程序设计》的数量（以本为单位）。每两个整数用一个空格分开。

输出:

输出一行，包含一个浮点数f，表示应付的总费用。精确到小数点后一位。

样例输入:

1 5 8 10 5 1 1 2 3 4

样例输出:

2140.2

【上机练习】

4.数组逆序重【1.6编程基础之一维数组04】

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:
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:
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

【上机练习】

5.年龄与疾病【1.6编程基础之一维数组05】

某医院想统计一下某项疾病的获得与否与年龄是否有关，需要对以前的诊断记录进行整理，按照0-18、19-35、36-60、61以上（含61）四个年龄段统计的患病人数占总患病人数的比例。

输入:
共2行，第一行为过往病人的数目n（0<n<=100)，第二行为每个病人患病时的年龄。
输出:
按照0-18、19-35、36-60、61以上(含61)四个年龄段输出该段患病人数占总患病人数的比例，以百分比的形式输出，精确到小数点后两位。每个年龄段占一行，共四行。
样例输入:
10
1 11 21 31 41 51 61 71 81 91
样例输出:
20.00%
20.00%
20.00%
40.00%

【上机练习】

6.校门外的树【1.6编程基础之一维数组06】Noip2005普及组第2题

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:

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。

对于20%的数据，区域之间没有重合的部分；对于其它的数据，区域之间有重合的情况。

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:

500 3
150 300
100 200
470 471

样例输出:

298

【上机练习】

7.向量点积计算【1.6编程基础之一维数组07】

在线性代数、计算几何中，向量点积是一种十分重要的运算。给定两个n维向量a=(a1,a2,...,an)和b=(b1,b2,...,bn)，求点积a·b=a1b1+a2b2+...+anbn。

输入:

第一行是一个整数n(1<=n<=1000)。

第二行包含n个整数a1,a2,...,an。

第三行包含n个整数b1,b2,...,bn。

相邻整数之间用单个空格隔开。每个整数的绝对值都不超过1000。

输出:

一个整数，即两个向量的点积结果。

样例输入:

3

1 4 6

2 1 5

样例输出:

36

【上机练习】

8.开关灯【1.5编程基础之循环控制28】

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

【上机练习】

9. 查找特定的值【1.9编程基础之顺序查找01】

在一个序列(下标从1开始)中查找一个给定的值，输出第一次出现的位置。

输入:
第一行包含一个正整数n，表示序列中元素个数。1 <=n<= 10000。
第二行包含n个整数，依次给出序列的每个元素，相邻两个整数之间用单个空格隔开。元素的绝对值不超过10000。
第三行包含一个整数x，为需要查找的特定值。x的绝对值不超过10000。

输出:
若序列中存在x，输出x第一次出现的下标；否则输出-1。

样例输入:
5
2 3 6 7 3
3
样例输出:
2

【上机练习】

9. 查找特定的值【1.9编程基础之顺序查找01】

在一个序列(下标从1开始)中查找一个给定的值，输出第一次出现的位置。

输入:
第一行包含一个正整数n，表示序列中元素个数。1 <=n<= 10000。
第二行包含n个整数，依次给出序列的每个元素，相邻两个整数之间用单个空格隔开。元素的绝对值不超过10000。
第三行包含一个整数x，为需要查找的特定值。x的绝对值不超过10000。

输出:
若序列中存在x，输出x第一次出现的下标；否则输出-1。

样例输入:
5
2 3 6 7 3
3
样例输出:
2

【上机练习】

10.不高兴的津津【1.9编程基础之顺序查找03】Noip2004普及组第1题

津津上初中了。妈妈认为津津应该更加用功学习，所以津津除了上学之外，还要参加妈妈为她报名的各科复习班。另外每周妈妈还会送她去学习朗诵、舞蹈和钢琴。但是津津如果一天上课超过八个小时就会不高兴，而且上得越久就会越不高兴。假设津津不会因为其它事不高兴，并且她的不高兴不会持续到第二天。请你帮忙检查一下津津下周的日程安排，看看下周她会不会不高兴；如果会的话，哪天最不高兴。

输入:

包括七行数据，分别表示周一到周日的日程安排。每行包括两个小于10的非负整数，用空格隔开，分别表示津津在学校上课的时间和妈妈安排她上课的时间。

输出:

包括一行，这一行只包含一个数字。如果不会不高兴则输出0，如果会则输出最不高兴的是周几（用1, 2, 3, 4, 5, 6, 7分别表示周一，周二，周三，周四，周五，周六，周日）。如果有两天或两天以上不高兴的程度相当，则输出时间最靠前的一天。

样例输入:	样例输出:
5 3 6 2 7 2 5 3 5 4 0 4 0 6	3

【上机练习】

11.最大值和最小值的差【1.9编程基础之顺序查找05】

输出一个整数序列中最大的数和最小的数的差。

输入:
第一行为M，表示整数个数，整数个数不会大于10000；
第二行为M个整数，以空格隔开，每个整数的绝对值不会大于10000。

输出:
输出M个数中最大值和最小值的差。

样例输入:
5
2 5 7 4 2

样例输出:
5

【上机练习】

12.不与最大数相同的数字之和【1.9编程基础之顺序查找07】

输出一个整数数列中不与最大数相同的数字之和。

输入:
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:
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

【上机练习】

13.白细胞计数【1.9编程基础之顺序查找08】

医院采样了某临床病例治疗期间的白细胞数量样本n份，用于分析某种新抗生素对该病例的治疗效果。为了降低分析误差，要先从这n份样本中去除一个数值最大的样本和一个数值最小的样本，然后将剩余n-2个有效样本的平均值作为分析指标。同时，为了观察该抗生素的疗效是否稳定，还要给出该平均值的误差，即所有有效样本（即不包括已扣除的两个样本）与该平均值之差的绝对值的最大值。

现在请你编写程序，根据提供的n个样本值，计算出该病例的平均白细胞数量和对应的误差。

输入:

输入的第一行是一个正整数n（2 <n<= 300），表明共有n个样本。

以下共有n行，每行为一个浮点数，为对应的白细胞数量，其单位为10^9/L。数与数之间以一个空格分开。

输出:

输出为两个浮点数，中间以一个空格分开。分别为平均白细胞数量和对应的误差，单位也是10^9/L。计算结果需保留到小数点后2位。

样例输入:

```
5
12.0
13.0
11.0
9.0
10.0
```

样例输出:

```
11.00 1.00
```

【上机练习】

14.直方图【1.9编程基础之顺序查找09】

给定一个非负整数数组，统计里面每一个数的出现次数。我们只统计到数组里最大的数。

假设 Fmax（Fmax<10000）是数组里最大的数，那么我们只统计{0,1,2.....Fmax}里每个数出现的次数。

输入:

第一行n是数组的大小。1 <= n <= 10000。

紧接着一行是数组的n个元素。

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，最大的数是3，因此我们只统计{0,1,2,3}的出现频数。

样例输入:

5

1 1 2 3 1

样例输出:

0

3

1

1

【上机练习】

15.最长平台【1.9编程基础之顺序查找12】

已知一个已经从小到大排序的数组，这个数组的一个平台（Plateau）就是连续的一串值相同的元素，并且这一串元素不能再延伸。例如，在 1，2，2，3，3，3，4，5，5，6中1，2-2，3-3-3，4，5-5，6都是平台。试编写一个程序，接收一个数组，把这个数组最长的平台找出 来。在上面的例子中3-3-3就是最长的平台。

输入:
第一行有一个整数n，为数组元素的个数。第二行有n个整数，整数之间以一个空格分开。

输出:
输出最长平台的长度。
样例输入:
10
1 2 2 3 3 3 4 5 5 6

样例输出:

3

【上机练习】

16.整数去重【1.9编程基础之顺序查找13】

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:
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100。
输出:
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

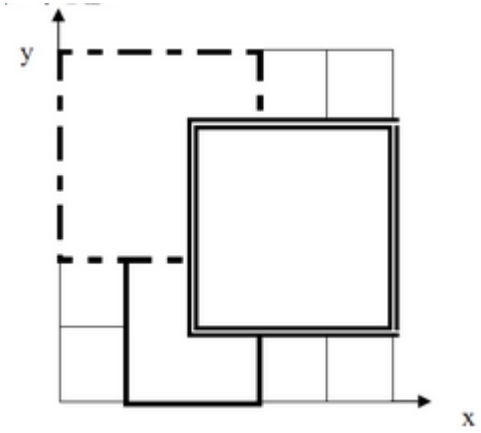
【上机练习】

17.铺地毯【1.9编程基础之顺序查找14】Noip2011提高组第1题

为了准备一个独特的颁奖典礼，组织者在会场的一片矩形区域（可看做是平面直角坐标系的第一象限）铺上一些矩形地毯。一共有n张地毯，编号从1到n。现在将这些地毯按照编号从小到大的顺序平行于坐标轴先后铺设，后铺的地毯覆盖在前面已经铺好的地毯之上。地毯铺设完成后，组织者想知道覆盖地面某个点的最上面的那张地毯的编号。注意：在矩形地毯边界和四个顶点上的点也算被地毯覆盖。

输入输出样例1说明：如下图，1号地毯用实线表示，2号地毯用虚线表示，3号用双实线表示，覆盖点（2，2）的最上面一张地毯是3号地毯。

输入输出样例2说明：如下图，1号地毯用实线表示，2号地毯用虚线表示，3号用双实线表示，覆盖点（4，5）的最上面一张地毯是3号地毯。



【上机练习】

输入:

第一行，一个整数n，表示总共有n张地毯。

接下来的n行中，第i+1行表示编号i的地毯的信息，包含四个正整数a，b，g，k，每两个整数之间用一个空格隔开，分别表示铺设地毯的左下角的坐标（a，b）以及地毯在x轴和y轴方向的长度。

第n+2行包含两个正整数x和y，表示所求的地面的点的坐标（x，y）。

输出:

输出共1行，一个整数，表示所求的地毯的编号；若此处没有被地毯覆盖则输出-1。

样例输入:

样例 #1：

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

样例 #2：

```
3
1 0 2 3
0 2 3 3
2 1 3 3
4 5
```

样例输出:

样例 #1：

```
3
```

样例 #2：

```
-1
```

第二节　二维数组

一、二维数组的定义

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。二维数组定义的一般格式：

数据类型 数组名[常量表达式1] [常量表达式2] ；

例如：int a[4][10];

a数组实质上是一个有4行、10列的表格，表格中可储存40个元素。第1行第1列对应a数组的a[0][0]，第n行第m列对应数组元素a[n-1][m-1]。

说明：当定义的数组下标有多个时，我们称为多维数组，下标的个数并不局限在一个或二个，可以任意多个，如定义一个三维数组a和四维数组b：

int a[100][3][5];

int b[100][100][3][5];

多维的数组引用赋值等操作与二维数组类似。

二、二维数组元素的引用

二维数组的数组元素引用与一维数组元素引用类似，区别在于二维数组元素的引用必须给出两个下标。

引用的格式为：

<数组名>[下标1][下标2]

说明：显然，每个下标表达式取值不应超出下标所指定的范围，否则会导致致命的越界错误。

例如,设有定义：int a[3][5];

则表示a是二维数组（相当于一个3*5的表格），共有3*5=15个元素，它们是：

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

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

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

因此可以看成一个矩阵（表格），a[2][3]即表示第3行第4列的元素。

三、二维数组的初始化

二维数组的初始化和一维数组类似。可以将每一行分开来写在各自的括号里，也可以把所有数据写在一个括号里。

例如：

```
int direct[4][2] = {{1,0},{0,1},{-1,0},{0,-1}}
int direct[4][2] = {1,0,0,1,-1,0,0,-1}              //尽量不要用
```

四、二维数组程序设计

例5.8 设有一程序

```
#include<cstdio>
#include<iostream>
#include<iomanip>
const int n=3;
using namespace std;
int a[n+1][n+1];
int main()
{
    for (int i=1; i<=n; ++i)
    {
      for (int j=1; j<=n; ++j)
        cin>>a[i][j];
    }
    for (int i=1; i<=n; ++i)
    {
        for (int j=1; j<=n; ++j)
            cout<<setw(5)<<a[j][i];
        cout<<endl;
    }
    return 0;
}
```

程序的输入:

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

程序的输出:

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

例5.9 已知一个6*6的矩阵（方阵），把矩阵二条对角线上的元素值加上10，然后输出这个新矩阵。

【分析】矩阵即表格，是一个二维数组，有6行6列共36个元素，每个矩阵都有二条对角线，本题难点在于对角线的元素怎么确定。

```
#include<iostream>
#include<iomanip>
using namespace std;
int a[7][7];
int main()
{
    for (int i=1; i<=6; ++i)                  //输入矩阵元素
      for (int j=1; j<=6; ++j)
        cin>>a[i][j];
    for (int i=1; i<=6; ++i)                  //更改对角线上元素的值
      for (int j=1; j<=6; ++j)
        if ((i==j)||(i+j==7)) a[i][j]+=10;    //寻找对角线的特征
    for (int i=1; i<=6; ++i)                  //输出6行6列的矩阵元素
      {
        for (int j=1; j<=6; ++j)
          cout<<setw(5)<<a[i][j];
        cout<<endl;
      }
     return 0;
}
```

例5.10　大部分元素是0的矩阵称为稀疏矩阵，假设有k个非0元素，则可把稀疏矩阵用K*3的矩阵简记之，其中第一列是行号，第二列是列号，第三列是该行、该列下的非元素的值。如：

```
0 0 0 5     写简记成：1 4 5     //第1行第4列有个数是5
0 2 0 0               2 2 2     //第2行第2列有个数是2
0 1 0 0               3 2 1     //第3行第2列有个数是1
```

试编程读入一稀疏矩阵，转换成简记形式，并输出。

【分析】 本题中需要解决的主要问题是查找非零元素并记忆位置。将原始矩阵存于数组a。转换后的矩阵存于数组b，当然b数组的行数可以控制在一个小范围内。

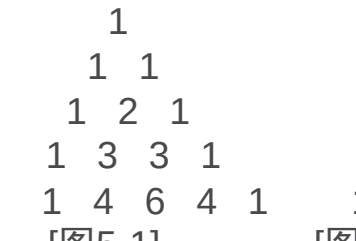
```
#include<iostream>
#include<iomanip>
const int n=3,m=5;
using namespace std;
int main()
{
    int    a[n+1][m+1],b[101][4],k=0;
    for (int i=1; i<=n; ++i)        //矩阵初始
      for (int j=1; j<=m; ++j)
        cin>>a[i][j];
    for (int i=1; i<=n; ++i)
      for (int j=1; j<=m; ++j)
        if (a[i][j]!=0)     //找到非零值，存储
        {
            ++k;
            b[k][1]=i;
            b[k][2]=j;
            b[k][3]=a[i][j];
         }
for (int i=1; i<=k; ++i)      //输出
    {
       for (int j=1; j<=3; ++j)
            cout<<setw(3)<<b[i][j];
       cout<<endl;
     }
    return 0;
}
```

运行结果：

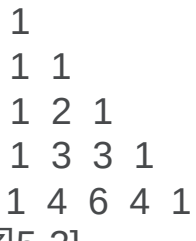
```
输入：0 0 0 0 5
      0 0 4 0 0
      1 0 0 0 1

输出：1 5 5
      2 3 4
      3 1 1
      3 5 1
```

例5.11 打印杨辉三角形的前10行。杨辉三角形如下图：



[图5-1]



[图5-2]

【问题分析】观察图5-1，大家不容易找到规律，但是如果将它转化为图5-2，不难发现杨辉三角形其实就是一个二维表的小三角形部分，假设通过二维数组yh存储，每行首尾元素为1，且其中任意一个非首位元素yh[i][j]的值其实就是yh[i-1][j-1]与yh[i-1][j]的和，另外每一行的元素个数刚好等于行数。有了数组元素的值，要打印杨辉三角形，只需要控制好输出起始位置就行了。

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
    int a[11][11];
        a[1][1]=1;                                          //设定第一行的值
    for (int i=2; i<=10; ++i)                               //从第二行开始推

        a[i][1]=1;  a[i][i]=1;                              //设定每一行的首尾值为1
        for (int j=2; j<=i-1; ++j)                          //当前行非首尾的数
          a[i][j]=a[i-1][j-1]+a[i-1][j];                    //每个数等于上一行的二个数之和
    }
    for (int i=1; i<=10; i++)
      {
        if (i!=10) cout<<setw(30-3*i)<<" ";                 //控制每行的起始位置，即空格数量
        for (int j=1; j<=i; j++) cout<<setw(6)<<a[i][j];
        cout<<endl;
      }
    return 0;
    }
```

例5.12 输入一串字符,字符个数不超过100,且以“.”结束。 判断它们是否构成回文。

【分析】所谓回文指从左到右和从右到左读一串字符的值是一样的，如12321,ABCBA,AA等。先读入要判断的一串字符（放入数组letter中），并记住这串字符的长度，然后首尾字符比较，并不断向中间靠拢，就可以判断出是否为回文。

程序如下：

```
#include<iostream>
using namespace std;
int main()
{    char ch,letter[101];
    int i=0,j=1;
    cout<<"Input a string:";
    cin>>ch;
    while (ch!='.')                                    //读入一个字符串以'.'号结束
    {
      ++i;
      letter[i]=ch;
      cin>>ch;
    }
    while ((j<i)&&(letter[j]==letter[i]))              //判断它是否是回文
    {
        --i; ++j;
    }
    if (j>=i) cout<<"Yes"<<endl;
       else   cout<<"No"<<endl;
    return 0;
}
```

例5.13 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。

【分析】：

类比数学中的矩阵，我们可以用一个所谓的二维数组来储存题目中的方阵。只需声明一个int a[MAXN][MAXN]，就可以获得一个大小为MAXN×MAXN的方阵。在声明时，两维的大小不必相同，因此也可以声明int a[30][50]这样的数组，第一维下标范围是0,1,2,…,29，第二维下标范围是0,1,2,…,49。

让我们从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。

```
#include<cstdio>
#include<cstring>
#define MAXN 10
int a[MAXN][MAXN];
int main()
{
        int n,x,y,tot=0;
        scanf("%d",&n);
        memset(a,0,sizeof(a));
        tot=a[x=0][y=n-1]=1;
        while (tot<n*n)
        {
        while (x+1<n && !a[x+1][y]) a[++x][y]=++tot;
        while (y-1>=0 && !a[x][y-1]) a[x][--y]=++tot;
        while (x-1>=0 && !a[x-1][y]) a[--x][y]=++tot;
        while (y+1<n && !a[x][y+1]) a[x][++y]=++tot;
        }
        for(x=0;x<n;++x)
        {
                for (y=0;y<n;++y) printf("%3d",a[x][y]);
                printf("\n");
        }
        return 0;
}
```

【说明】：

这段程序充分利用了C++语言简洁的优势。首先，赋值x=0和y=n-1后马上要把它们作为a数组的下标，因此可以合并完成；tot和a[0][n-1]都要赋值1，也可以合并完成。这样，我们用一条语句完成了多件事情，而且并没有牺牲程序的可读性，这段代码的含义显而易见。

那4条while语句有些难懂，不过十分相似，因此只需介绍其中的第一条：不断向下走，并且填数。我们的原则是：先判断，再移动，而不是走一步以后发现越界了再退回来。这样，我们需要进行“预判”，即是否越界，以及如果继续往下走会不会到达一个已经填过的格子。越界只需判断x+1<n，因为y值并没有修改；下一个格子是(x+1,y)，因此只需a[x+1][y]==0，简写成!a[x+1][y]（其中!是“逻辑非”运算符）。

细心的读者也许会发现这里的一个“潜在bug”；如果越界，x+1会等于n，a[x+1][y]将访问非法内存！幸运的是，这样的担心是不必要的。&&是短路运算符。如果x+1<n为假，将不会计算!a[x+1][y]，也就不会越界了。

至于为什么是++tot而不是tot++，留给读者思考。

【上机练习】

1.矩阵交换行【1.8编程基础之多维数组01】

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

【上机练习】

2.同行列对角线的格【1.8编程基础之多维数组02】

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，如下图：

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

【上机练习】

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)

【上机练习】

3.计算矩阵边缘元素之和【1.8编程基础之多维数组03】

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。

输入:
第一行分别为矩阵的行数m和列数n（m<100，n<100），两者之间以一个空格分开。
接下来输入的m行数据中，每行包含n个整数，整数之间以一个空格分开。
输出:
输出对应矩阵的边缘元素和。

样例输入:
3 3
3 4 1
3 7 1
2 0 1
样例输出:
15

【上机练习】

4.计算鞍点【1.8编程基础之多维数组05】

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。

例如：在下面的例子中（第4行第1列的元素就是鞍点，值为8 ）。

输入:

输入包含一个5行5列的矩阵。

输出:

如果存在鞍点，输出鞍点所在的行、列及其值，如果不存在，输出"not found"。

样例输入:

11 3 5 6 9
12 4 7 8 10
10 5 6 9 11
8 6 4 7 2
15 10 11 20 25

样例输出:

4 1 8

【上机练习】

5.图像相似度【1.8编程基础之多维数组06】

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
```

【上机练习】

6.矩阵加法【1.8编程基础之多维数组07】

输入两个n行m列的矩阵A和B，输出它们的和A+B。

输入:
第一行包含两个整数n和m，表示矩阵的行数和列数(1<=n<=100，1<=m<=100)。
接下来n行，每行m个整数，表示矩阵A的元素。
接下来n行，每行m个整数，表示矩阵B的元素。
相邻两个整数之间用单个空格隔开，每个元素均在1~1000之间。
输出:
n行，每行m个整数，表示矩阵加法的结果。相邻两个整数之间用单个空格隔开。

样例输入:

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

样例输出:

```
2 4 6
5 7 9
8 10 12
```

【上机练习】

7.矩阵乘法【1.8编程基础之多维数组08】

计算两个矩阵的乘法。n*m阶的矩阵A乘以m*k阶的矩阵B得到的矩阵C 是n*k阶的，且C[i][j] = A[i][0]*B[0][j] + A[i][1]*B[1][j] + …… +A[i][m-1]*B[m-1][j](C[i][j]表示C矩阵中第i行第j列元素)。

输入:

第一行为n, m, k，表示A矩阵是n行m列，B矩阵是m行k列，n, m, k均小于100。

然后先后输入A和B两个矩阵，A矩阵n行m列，B矩阵m行k列，矩阵中每个元素的绝对值不会大于1000。

输出:

输出矩阵C，一共n行，每行k个整数，整数之间以一个空格分开。

样例输入:

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

样例输出:

```
2 2 2
2 2 2
2 2 2
```

【上机练习】

8.矩阵转置【1.8编程基础之多维数组09】

输入一个n行m列的矩阵A，输出它的转置AT。

输入:
第一行包含两个整数n和m，表示矩阵A的行数和列数(1<=n<=100，1<=m<=100)。
接下来n行，每行m个整数，表示矩阵A的元素。相邻两个整数之间用单个空格隔开，每个元素均在1~1000之间。
输出:
m行，每行n个整数，为矩阵A的转置。相邻两个整数之间用单个空格隔开。
样例输入:
3 3
1 2 3
4 5 6
7 8 9
样例输出:
1 4 7
2 5 8
3 6 9

【上机练习】

9.图像旋转【1.8编程基础之多维数组10】

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。

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，为顺时针旋转90度后的图像。相邻两个整数用单个空格隔开。
样例输入:
3 3
1 2 3
4 5 6
7 8 9
样例输出:
7 4 1
8 5 2
9 6 3

【上机练习】

10.图像模糊处理【1.8编程基础之多维数组12】

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:

```
4 5
100 0 100 0 50
50 100 200 0 0
50 50 100 100 200
100 100 50 50 100
```

样例输出:

```
100 0 100 0 50
50 80 100 60 0
```

第三节　字符数组和字符串类型

无论数组的下标有几个，类型如何，但数组中全体元素的类型必须相同。数组元素的类型可以是任何类型，当它是字符型时，我们称它为字符数组。由于字符数组与字符类型的应用是计算机非数值处理的重要方面之一，所以我们把它们两个放在一起进行讨论。

下面我们举例说明字符数组的应用。

一、字符类型

字符类型为由一个字符组成的字符常量或字符变量。

字符常量定义:

const

字符常量=‘字符’

字符变量定义:

char 字符变量;

字符类型是一个有序类型, 字符的大小顺序按其ASCII代码的大小而定。

例5.14 按字母表顺序和逆序每隔一个字母打印。即打印出：

a c e g i k m o q s u w y

z x r v t p n l j h f d b

```
#include<iostream>
 #include<iomanip>
 using namespace std;
 int main()
 {
    for (char letter='a'; letter<='z'; letter+=2)
      cout<<setw(3)<<letter;
    cout<<endl;
    for (char letter='z'; letter>='a'; letter-=2)
      cout<<setw(3)<<letter;
    return 0;
 }
```

【说明】程序中，我们利用了字符类型是顺序类型这一特性，灵活利用字符变量当作循环变量，使程序处理起来比较直观。

二、字符数组

字符数组是指元素为字符的数组。字符数组是用来存放字符序列或字符串的。字符数组也有一维、二维和三维之分。

1、字符数组的定义格式

字符数组定义格式同于一般数组，所不同的是数组类型是字符型，第一个元素同样是从ch1[0]开始，而不是ch1[1]。具体格式如下：

[存储类型] char 数组名[常量表达式1]…

例如：

```
char ch1[5];            //数组ch1是一个具有5个字符元素的一维字符数组
char ch2[3][5];         //数组ch2是一个具有15个字符元素的二维字符数组
```

2.字符数组的赋值

字符数组赋值类似于一维数组，赋值分为数组的初始化和数组元素的赋值。初始化的方式有用字符初始化和用字符串初始化两种，也有用初始值表进行初始化的。

(1).用字符初始化数组

例如：

char chr1[5]={‘a’，‘b’，‘c’，‘d’，‘e’};

初始值表中的每个数据项是一个字符，用字符给数组chr1的各个元素初始化。当初始值个数少于元素个数时，从首元素开始赋值，剩余元素默认为空字符。

字符数组中也可以存放若干个字符，也可以来存放字符串。两者的区别是字符串有一结束符(‘\0’)。反过来说，在一维字符数组中存放着带有结束符的若干个字符称为字符串。字符串是一维数组，但是一维字符数组不等于字符串。

例如：

char chr2[5]={‘a’，‘b’，‘c’，‘d’，‘\0’}; 即在数组chr2中存放着一个字符串“abcd”。

(2).用字符串初始化数组

用一个字符串初始化一个一维字符数组，可以写成下列形式：

char chr2[5]=“abcd”;

使用此格式均要注意字符串的长度应小于字符数组的大小或等于字符数组的大小减1。同理，对二维字符数组来讲，可存放若干个字符串。可使用由若干个字符串组成的初始值表给二维字符数组初始化。

例如：char chr3[3][4]={“abc”，“mno”，“xyz”};在数组ch3中存放3个字符串，每个字符串的长度不得大于3。

(3).数组元素赋值

字符数组的赋值是给该字符数组的各个元素赋一个字符值。

例如：

char chr[3];

chr[0]=‘a’; chr[1]=‘b’;chr[2]=‘c’;

对二维、三维字符数组也是如此。当需要将一个数组的全部元素值赋予另一数组时，不可以用数组名直接赋值的方式，要使用字符串拷贝函数来完成。

(4).字符常量和字符串常量的区别

①两者的定界符不同，字符常量由单引号括起来，字符串常量由双引号括起来。

②字符常量只能是单个字符，字符串常量则可以是多个字符。

③可以把一个字符常量赋给一个字符变量，但不能把一个字符串常量赋给一个字符变量。

④字符常量占一个字节，而字符串常量占用字节数等于字符串的字节数加1。增加的一个字节中存放字符串结束标志‘\0’。例如：字符常量‘a’占一个字节，字符串常量“a”占二个字节。

三、字符串的输入与输出

字符串可以作为一维字符数组来处理，那么字符串的输入和输出也可以按照数组元素来处理，本节不再做介绍。本节仅介绍将字符串作为一个整体进行输入和输出的语句。

1、输入

从键盘输入一个字符数组可以使用scanf语句或gets语句。

(1)scanf语句

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：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。

(2)gets语句

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!

2、输出

向屏幕输出一个字符串可以使用printf语句或puts语句。

(1)printf语句

格式：printf(“%s”,字符串名称);

说明：

①用%s格式输出时，printf的输出项只能是字符串(字符数组)名称，而不能是数组元素。例如：printf(“%s”,a[5]);是错误的。

②输出字符串不包括字符串结束标志符‘\0’。

(2) puts语句

格式：puts(字符串名称);

说明：puts语句输出一个字符串和一个换行符。对于已经声明过的字符串a，printf(“%s\n”,a)和 puts(a)是等价的。

例5.15 C++中，一个字符串中的字符可以通过其对应的下标灵活使用。

```
#include<cstdio>          // gets()调用cstdio库
#include<iostream>
#include<cstring>         //strlen()调用cstring库, 调用string库在高版C++下编译出错
using namespace std;
int main()
{
    char st[100];
    gets(st);                  //gets为专门读字符串的
                               函数, 读取一行字符串
    for (int i=0; i<strlen(st); ++i)
                               //输出st串中的第i个字符
      cout<<st[i];
    return 0;
}
```

例5.16 对给定的10个国家名，按其字母的顺序输出。

【参考程序1】

```
#include<cstdio>
#include<iostream>
#include<cstring>
using namespace std;
int main()
{   char t[21],cname[11][21];
  for (int i=1; i<=10; ++i) gets(cname[i]);
                                        //gets为专门读字符串的函数, 读取一行字符串
   for (int i=1; i<=9; ++i)
   {   int k=i;
       for (int j=i+1; j<=10; ++j)
         if (strcmp(cname[k],cname[j])>0) k=j;
       strcpy(t,cname[i]);
       strcpy(cname[i],cname[k]);
       strcpy(cname[k],t);
   }
   for (int i=1; i<=10; ++i) cout<<cname[i]<<endl;
   return 0;
}
```

【参考程序2】（详见第八章第一节和第三节）

```
#include<algorithm>
#include<iostream>
#include<string>
using namespace std;
string cname[10];
int main(){
 for (int i=0;i!=10;++i) getline(cin,cname[i]);
 sort(cname,cname+10);                    //利用C++库函数排序
 for (int i=0;i!=10;++i) cout<<cname[i]<<endl;
 return 0;
}
```

三、字符串处理函数

系统提供了一些字符串处理函数，用来为用户提供一些字符串的运算。常用的字符串函数介绍如下。

函数格式	函数功能
strcat(字符串名1,字符串名2)	将字符串2连接到字符串1后边，返回字符串1的值。
strncat(字符串名1,字符串名2,长度n)	将字符串2前n个字符连接到字符串1后边，返回字符串1的值。
strcpy(字符串名1,字符串名2)	将字符串2复制到字符串1后边，返回字符串1的值。
strncpy(字符串名1,字符串名2,长度n)	将字符串2前n个字符复制到字符串1后边，返回字符串1的值。
strcmp(字符串名1,字符串名2)	比较字符串1和字符串2的大小，比较的结果由函数带回； 如果字符串1>字符串2，返回一个正整数； 如果字符串1=字符串2，返回0； 如果字符串1<字符串2，返回一个负整数；
strncmp(字符串名1,字符串名2,长度n)	比较字符串1和字符串2的前n个字符进行比较，函数返回值的情况同strcmp函数；
strlen(字符串名)	计算字符串的长度，终止符’\0’不算在长度之内
strlwr(字符串名)	将字符串中大写字母换成小写字母
strupr(字符串名)	将字符串中小写字母换成大写字母

四、应用举例

例5.17 数字统计(Noip2010)

【问题描述】

请统计某个给定范围[L, R]的所有整数中，数字2 出现的次数。

比如给定范围[2, 22]，数字2 在数2 中出现了1 次，在数12 中出现1 次，在数20 中出现1 次，在数21 中出现1 次，在数22 中出现2 次，所以数字2 在该范围内一共出现了6 次。

【输入】

输入文件名为two.in 。

输入共1 行，为两个正整数L 和R，之间用一个空格隔开。

【输出】

输出文件名为two.out 。

输出共1 行，表示数字2 出现的次数。

【输入样例1】two.in

2 22

【输出样例1】two.out

6

【输入样例2】two.in

2 100

【输出样例2】two.out

20

【数据范围】

1 ≤L ≤R ≤10000

【算法分析1】

枚举[L,R]区间的所有整数，对于每个整数x：

1.将整数x转化成字符串s，可以用sprintf(s,"%d",x)来实现；

2.枚举字符串s的每个字符判断是否为2。

【参考程序1】

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
char s[10];
int main()
{
        int l,r,ans=0;
        cin>>l>>r;
        for (int i=l; i<=r; ++i)
        {
            sprintf(s,"%d",i);
            l=strlen(s);
            for (int j=0; j<=l-1; ++j)
                if (s[j]=='2') ++ans;
        }
        cout<<ans;
        return 0;
}
```

【算法分析2】

枚举[L,R]区间的所有整数，对于每个整数x：

先判断x的最后一位是否为2（即 x%10==2），然后将x的最后一位删除（即 x/=10），循环操作，直到x值为0。

【参考程序2】

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
int main()
{
	int l,r,ans=0;
	cin>>l>>r;
	for (int i=l; i<=r; ++i)
	{
		int x=i;
		while (x>0)
		{
			if (x%10==2) ++ans;
			x/=10;
		}
	}
	cout<<ans;
	return 0;
}
```

例5.18 数字反转(Noip2011)

【问题描述】

给定一个整数，请将该数各个位上数字反转得到一个新数。新数也应满足整数的常见形式，即除非给定的原数为零，否则反转后得到的新数的最高位数字不应为零（参见样例2）。

【输入】

输入文件名为reverse.in。

输入共1 行，一个整数N。

【输出】

输出文件名为reverse.out。

输出共1 行，一个整数，表示反转后的新数。

【输入样例1】

123

【输出样例1】

321

【输入样例2】

-380

【输出样例2】

-83

【数据范围】

-1,000,000,000 ≤ N≤ 1,000,000,000。

【算法分析1】

1.将整数N转化成字符串s，可以用sprintf(s,"%d",N)来实现；

2.对字符串进行反转操作；

3.将字符串s转换成数字N,可以用sscanf(s,"%d",&N)来实现。

4.输出数字N。

【参考程序1】

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;
char s[100],c[100];
int main()
{
    int n,l;
    cin>>n;
    sprintf(s,"%d",n);
    l=strlen(s);
    for (int i=0; i<=l-1; ++i)  c[l-i-1]=s[i];
    if (n<0) cout<<"-";
    sscanf(c,"%d",&n);
    cout<<n;
    return 0;
}
```

【算法分析2】

判断数字是否为负数，如果是则先输出符号，并将数字取绝对值；

将数字从后往前转换成字符串；

去掉前导0，然后输出数字。

【参考程序2】

```
#include<iostream>
#include<cstdio>
using namespace std;
string s;
void Solve(int x)       //数字从后往前转换成字符串
{
    while (x>0) { s+=x%10+48; x/=10; }
}
int main()
{
    int n;
    scanf("%d",&n);
    if (n<0) { cout<<"-"; n=-n;}
    Solve(n);
    int j=0,len=s.size();
    while ( (j+1!=len) && (s[j]=='0')) ++j;       //去掉前导0
    for (;j<len; ++j) cout<<s[j];
    return 0;
}
```

例5.19 统计单词数(Noip2011)

【问题描述】

一般的文本编辑器都有查找单词的功能，该功能可以快速定位特定单词在文章中的位置，有的还能统计出特定单词在文章中出现的次数。

现在，请你编程实现这一功能，具体要求是：给定一个单词，请你输出它在给定的文章中出现的次数和第一次出现的位置。注意：匹配单词时，不区分大小写，但要求完全匹配，即给定单词必须与文章中的某一独立单词在不区分大小写的情况下完全相同（参见样例1），如果给定单词仅是文章中某一单词的一部分则不算匹配（参见样例2）。

【输入】

输入文件名为stat.in，2 行。

第1 行为一个字符串，其中只含字母，表示给定单词；

第2 行为一个字符串，其中只可能包含字母和空格，表示给定的文章。

【输出】

输出文件名为stat.out。

只有一行，如果在文章中找到给定单词则输出两个整数，两个整数之间用一个空格隔开，分别是单词在文章中出现的次数和第一次出现的位置（即在文章中第一次出现时，单词首字母在文章中的位置，位置从0 开始）；如果单词在文章中没有出现，则直接输出一个整数-1。

【输入样例1】

To

to be or not to be is a question

【输出样例1】

2 0

【输入输出样例1说明】

输出结果表示给定的单词To 在文章中出现两次，第一次出现的位置为0。

【数据范围】

1 ≤ 单词长度≤ 10。

1 ≤ 文章长度≤ 1,000,000。

【算法分析】

枚举文章中的每个单词；

判断两个单词长度是否相同；

枚举单词中的每个字母，判断是否都相同，如果都相同则答案加一。

【参考程序】

```
#include<cstdio>
#include<cstring>
#include<iostream>
using namespace std;
int main()
{
    int i,j,t=0,tt=0;
    char s[1000001],ss[11];
    cin.getline(ss,11);
    cin.getline(s,1000001);
    for(i=0;i<=strlen(s)-strlen(ss);++i)
    {
       for (j=0;j<=strlen(ss)-1;++j)
       {
          if (toupper(s[j+i])!=toupper(ss[j])) break;
          if (i>0&&s[i-1]!=' ') break;
        }
       if (j==strlen(ss)&&(s[j+i]==' '||j+i==strlen(s)))
           {t++;if (t==1) tt=i;}
    }
    if (t==0) printf("-1");
       else  printf("%d %d\n",t,tt);
     return 0;
}
```

【算法分析2】（详见第八章第三节）

1.先预处理将所有字母转为大写,将pos置为0；

2.利用string类的find(word,pos)函数查找是否存在单词；

3.如果存在，累加答案，更新pos，返回第二步，否则退出。

【参考程序2】

```
//string::size_type 为 string 定义的无符整形
#include<algorithm>
#include<iostream>
#include<string>
#include<cctype>
using namespace std;
string word,str;
```

```
int main(){
    int first=0,ans=0;
    cin>>word; word=" "+word+" ";
    for (string::size_type i=0;i!=word.size();++i)//string::size_type为无符类型
        word[i]=toupper(word[i]);
    getline(cin,str);                    //当我们读完word后，还有换行符
    getline(cin,str); str=" "+str+" ";
    for (string::size_type i=0;i!=str.size();++i)
        str[i]=toupper(str[i]);
    for (string::size_type i=str.find(word);i<str.size();
                i=str.find(word,i+word.size()-1),++ans) {
        if (!ans) first=i;                    //如果是第一次找到，记录位置
    }
    if (ans) cout<<ans<<' '<<first<<endl;
    else cout<<-1<<endl;
    return 0;
}
```

【上机练习】

1.统计数字字符个数【1.7编程基础之字符串01】

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:

4

2.找第一个只出现一次的字符【1.7编程基础之字符串02】

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100000。

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

50 80 100 90 200

100 100 50 50 100

【上机练习】

3.基因相关性【1.7编程基础之字符串03】

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

【上机练习】

4.石头剪子布【1.7编程基础之字符串04】

石头剪子布，是一种猜拳游戏。起源于中国，然后传到日本、朝鲜等地，随着亚欧贸易的不断发展它传到了欧洲，到了近现代逐渐风靡世界。简单明了的规则，使得石头剪子布没有任何规则漏洞可钻，单次玩法比拼运气，多回合玩法比拼心理博弈，使得石头剪子布这个古老的游戏同时用于“意外”与“技术”两种特性，深受世界人民喜爱。

游戏规则：石头打剪刀，布包石头，剪刀剪布。

现在，需要你写一个程序来判断石头剪子布游戏的结果。

输入:

第一行是一个整数N，表示一共进行了N次游戏。1 <= N <= 100。

接下来N行的每一行包括两个字符串，表示游戏参与者Player1，Player2的选择（石头、剪子或者是布）：

S1 S2

字符串之间以空格隔开S1,S2只可能取值在{"Rock", "Scissors", "Paper"}(大小写敏感)中。

输出:

输出包括N行，每一行对应一个胜利者（Player1或者Player2），或者游戏出现平局，则输出Tie。

样例输入:

```
3
Rock Scissors
Paper Paper
Rock Paper
```

样例输出:

```
Player1
Tie
Player2
```

【上机练习】

5.输出亲朋字符串【1.7编程基础之字符串05】

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

【上机练习】

6.合法C标识符查【1.7编程基础之字符串06】

给定一个不包含空白符的字符串，请判断是否是C语言合法的标识符号(注：题目保证这些字符串一定不是C语言的保留字)。

C语言标识符要求：

1.非保留字；

2.只包含字母、数字及下划线（“_”）。

3.不以数字开头。

输入:

一行，包含一个字符串，字符串中不包含任何空白字符，且长度不大于20。

输出:

一行，如果它是C语言的合法标识符，则输出yes，否则输出no。

样例输入:

RKPEGX9R;TWyYcp

样例输出:

no

【上机练习】

7.配对碱基链【1.7编程基础之字符串07】

脱氧核糖核酸(DNA)由两条互补的碱基链以双螺旋的方式结合而成。而构成DNA的碱基共有4种，分别为腺瞟呤(A)、鸟嘌呤(G)、胸腺嘧啶(T)和胞嘧啶(C)。我们知道，在两条互补碱基链的对应位置上，腺瞟呤总是和胸腺嘧啶配对，鸟嘌呤总是和胞嘧啶配对。你的任务就是根据一条单链上的碱基序列，给出对应的互补链上的碱基序列。

输入：

一个字符串，表示一条碱基链。这个字符串只含有大写字母A、T、G、C，分别表示腺瞟呤、胸腺嘧啶、鸟嘌呤和胞嘧啶。字符串长度不超过255。

输出：

一个只含有大写字母A、T、G、C的字符串，为与输入的碱基链互补的碱基链。

样例输入:

ATATGGATGGTGTTTGGCTCTG

样例输出:

TATACCTACCACAAACCGAGAC

【上机练习】

8.密码翻译【1.7编程基础之字符串08】

在情报传递过程中，为了防止情报被截获，往往需要对情报用一定的方式加密，简单的加密算法虽然不足以完全避免情报被破译，但仍然能防止情报被轻易的识别。我们给出一种最简的的加密方法，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。

输入：

输入一行，包含一个字符串，长度小于80个字符。

输出：

输出每行字符串的加密字符串。

样例输入:

Hello! How are you!

样例输出:

Ifmmp! Ipx bsf zpv!

【上机练习】

9.加密的病历单【1.7编程基础之字符串10】

小英是药学专业大三的学生，暑假期间获得了去医院药房实习的机会。

在药房实习期间，小英扎实的专业基础获得了医生的一致好评，得知小英在计算概论中取得过好成绩后，主任又额外交给她一项任务，解密抗战时期被加密过的一些伤员的名单。

经过研究，小英发现了如下加密规律(括号中是一个“原文 -> 密文”的例子)
1.原文中所有的字符都在字母表中被循环左移了三个位置（dec -> abz）
2.逆序存储（abcd -> dcba ）
3.大小写反转（abXY -> ABxy）

输入:
一个加密的字符串。(长度小于50且只包含大小写字母)
输出:
输出解密后的字符串。

样例输入:
GSOOWFASOq
样例输出:
Trvdizrrvj

【上机练习】

10.将字符串中的小写字母转换成大写字母【1.7编程基础之字符串11】

给定一个字符串，将其中所有的小写字母转换成大写字母。

输入:

输入一行，包含一个字符串（长度不超过100，可能包含空格）。

输出:

输出转换后的字符串。

样例输入:

helloworld123Ha

样例输出:

HELLOWORLD123HA

【上机练习】

11.大小写字母互换【1.7编程基础之字符串12】

把一个字符串中所有出现的大写字母都替换成小写字母，同时把小写字母替换成大写字母。

输入:
输入一行：待互换的字符串。

输出:
输出一行：完成互换的字符串(字符串长度小于80)。

样例输入:
If so, you already have a Google Account. You can sign in on the right.

样例输出:
iF SO, YOU ALREADY HAVE A gOOGLE aCCOUNT. yOU CAN SIGN IN ON THE RIGHT.

【上机练习】

12.整理药名【1.7编程基础之字符串13】

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如果是字母要大写，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:

```
4
AspiRin
cisapride
2-PENICILLIN
Cefradine-6
```

样例输出:

```
Aspirin
Cisapride
2-penicillin
Cefradine-6
```

【上机练习】

13.忽略大小写的字符串比较【1.7编程基础之字符串14】

一般我们用strcmp可比较两个字符串的大小，比较方法为对两个字符串从前往后逐个字符相比较（按ASCII码值大小比较），直到出现不同的字符或遇到'\0'为止。如果全部字符都相同，则认为相同；如果出现不相同的字符，则以第一个不相同的字符的比较结果为准（注意：如果某个字符串遇到'\0'而另一个字符串还未遇到'\0'，则前者小于后者）。但在有些时候，我们比较字符串的大小时，希望忽略字母的大小，例如"Hello"和"hello"在忽略字母大小写时是相等的。请写一个程序，实现对两个字符串进行忽略字母大小写的大小比较。

输入:

输入为两行，每行一个字符串，共两个字符串。（每个字符串长度都小于80）

输出:

如果第一个字符串比第二个字符串小，输出一个字符"<";

如果第一个字符串比第二个字符串大，输出一个字符">";

如果两个字符串相等，输出一个字符"="。

样例输入:

Hello, how are you?

hello, How are you?

样例输出:

=

【上机练习】

14.验证子串【1.7编程基础之字符串15】

输入两个字符串，验证其中一个串是否为另一个串的子串。

输入:
输入两个字符串， 每个字符串占一行，长度不超过200且不含空格。

输出:
若第一个串s1是第二个串s2的子串，则输出(s1) is substring of (s2)
否则，若第二个串s2是第一个串s1的子串，输出(s2) is substring of (s1)
否则，输出 No substring。

样例输入:
abc
dddncabca

样例输出:
abc is substring of dddncabca

【上机练习】

15.删除单词后缀【1.7编程基础之字符串16】

给定一个单词，如果该单词以er、ly或者ing后缀结尾， 则删除该后缀（题目保证删除后缀后的单词长度不为0）， 否则不进行任何操作。

输入:

输入一行，包含一个单词（单词中间没有空格，每个单词最大长度为32）。

输出:

输出按照题目要求处理后的单词。

样例输入:

referer

样例输出:

refer

【上机练习】

16.过滤多余的空格【1.7编程基础之字符串19】

一个句子中也许有多个连续空格，过滤掉多余的空格，只留下一个空格。

输入:

一行，一个字符串（长度不超过200），句子的头和尾都没有空格。

输出:

过滤之后的句子。

样例输入:

Hello world.This is c language.

样例输出:

Hello world.This is c language.

【上机练习】

17.单词的长度【1.7编程基础之字符串20】

输入一行单词序列，相邻单词之间由1个或多个空格间隔，请计算各个单词的长度。

注意:如果有标点符号(如连字符，逗号)，标点符号算作与之相连的词的一部分。没有被空格间开的符号串，都算作单词。

输入:

一行单词序列，最少1个单词，最多300个单词，单词之间用至少1个空格间隔。单词序列总长度不超过1000。

输出:

依次输出对应单词的长度，之间以逗号间隔。

【上机练习】

18.最长最短单词【1.7编程基础之字符串21】

输入1行句子(不多于200个单词，每个单词长度不超过100)，只包含字母、空格和逗号。单词由至少一个连续的字母构成，空格和逗号都是单词间的间隔。

试输出第1个最长的单词和第1个最短单词。

输入:
一行句子。

输出:
第1行，第一个最长的单词。
第2行，第一个最短的单词。

样例输入:
I am studying Programming language C in Peking University
样例输出:
Programming
I

提示:如果所有单词长度相同，那么第一个单词既是最长单词也是最短单词。

【上机练习】

19.单词翻转【1.7编程基础之字符串23】

输入一个句子(一行)，将句子中的每一个单词翻转后输出。

输入:

只有一行，为一个字符串，不超过500个字符。单词之间以空格隔开。

输出:

翻转每一个单词后的字符串，单词之间的空格需与原文一致。

样例输入:

hello world

样例输出:

olleh dlrow

【上机练习】

20.字符串p型编码【1.7编程基础之字符串26】

给定一个完全由数字字符（'0','1','2',…,'9'）构成的字符串str，请写出str的p型编码串。例如：字符串122344111可被描述为"1个1、2个2、1个3、2个4、3个1"，因此我们说122344111的p型编码串为1122132431；类似的道理，编码串101可以用来描述1111111111；00000000000可描述为"11个0"，因此它的p型编码串即为110；100200300可描述为"1个1、2个 0、1个2、2个0、1个3、2个0"，因此它的p型编码串为112012201320。

输入:

输入仅一行，包含字符串str。每一行字符串最多包含1000个数字字符。

输出:

输出该字符串对应的p型编码串。

样例输入:

122344111

样例输出:

1122132431

【上机练习】

21.判断字符串是否为回文【1.7编程基础之字符串28】

输入一个字符串，输出该字符串是否回文。回文是指顺读和倒读都一样的字符串。

输入:

输入为一行字符串（字符串中没有空白字符，字符串长度不超过100）。

输出:

如果字符串是回文，输出yes；否则，输出no。

样例输入:

abcdedcba

样例输出:

yes

【上机练习】

22.最高分数的学生姓名【1.9编程基础之顺序查找02】

输入学生的人数，然后再输入每位学生的分数和姓名，求获得最高分数的学生的姓名。

输入:

第一行输入一个正整数N（N <= 100），表示学生人数。接着输入N行，每行格式如下：

分数 姓名

分数是一个非负整数，且小于等于100；

姓名为一个连续的字符串，中间没有空格，长度不超过20。

数据保证最高分只有一位同学。

输出:

获得最高分数同学的姓名。

样例输入:

5

87 lilei

99 hanmeimei

97 lily

96 lucy

77 jim

样例输出:

hanmeimei

【上机练习】

23.谁拿了最多奖学金【1.9编程基础之顺序查找04】Noip2005提高组第1题

某校的惯例是在每学期的期末考试之后发放奖学金。发放的奖学金共有五种，获取的条件各自不同：

1)院士奖学金，每人8000元，期末平均成绩高于80分（>80），并且在本学期内发表1篇或1篇以上论文的学生均可获得；

2)五四奖学金，每人4000元，期末平均成绩高于85分（>85），并且班级评议成绩高于80分（>80）的学生均可获得；

3)成绩优秀奖，每人2000元，期末平均成绩高于90分（>90）的学生均可获得；

4)西部奖学金，每人1000元，期末平均成绩高于85分（>85）的西部省份学生均可获得；

5)班级贡献奖，每人850元，班级评议成绩高于80分（>80）的学生干部均可获得；

只要符合条件就可以得奖，每项奖学金的获奖人数没有限制，每名学生也可以同时获得多项奖学金。例如姚林的期末平均成绩是87分，班级评议成绩82分，同时他还是一位学生干部，那么他可以同时获得五四奖学金和班级贡献奖，奖金总数是4850元。

现在给出若干学生的相关数据，请计算哪些同学获得的奖金总数最高（假设总有同学能满足获得奖学金的条件）。

输入:

第一行是一个整数N（1 <= N <= 100），表示学生的总数。接下来的N行每行是一位学生的数据，从左向右依次是姓名，期末平均成绩，班级评议成绩，是否是学生干部，是否是西部省份学生，以及发表的论文数。姓名是由大小写英文字母组成的长度不超过20的字符串（不含空格）；期末平均成绩和班级评议成绩都是0到100之间的整数（包括0和100）；是否是学生干部和是否是西部省份学生分别用一个字符表示，Y表示是，N表

【上机练习】

示不是；发表的论文数是0到10的整数（包括0和10）。每两个相邻数据项之间用一个空格分隔。

输出:

包括三行，第一行是获得最多奖金的学生的姓名，第二行是这名学生获得的奖金总数。如果有两位或两位以上的学生获得的奖金最多，输出他们之中在输入文件中出现最早的学生的姓名。第三行是这N个学生获得的奖学金的总数。

样例输入:

```
4
YaoLin 87 82 Y N 0
ChenRuiyi 88 78 N Y 1
LiXin 92 88 N N 0
ZhangQin 83 87 Y N 1
```

样例输出:

```
ChenRuiyi
9000
28700
```

【上机练习】

24.连续出现的字符【1.9编程基础之顺序查找11】

给定一个字符串，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1000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

【上机练习】

25. 最长单词2【1.13编程基础之综合应用16】

一个以'.'结尾的简单英文句子，单词之间用空格分隔，没有缩写形式和其它特殊形式。

输入：

一个以'.'结尾的简单英文句子（长度不超过500），单词之间用空格分隔，没有缩写形式和其它特殊形式

输出：

该句子中最长的单词。如果多于一个，则输出第一个

样例输入：

I am a student of Peking University.

样例输出：

University