

第六章 数组

一维数组的定义和引用

数组是同一类型的一组值（10个 char 或15个 int），在内存中顺序存放。

整个数组共用一个名字，而其中的每一项又称为一个元素。

一、定义方式：

类型说明符 数组名[常量表达式];

定义类型

元素个数

int a[4]; // 表明a数组由4个int型元素组成

数组名称

必须是常数

int a[4]; // 表明a数组由4个int型元素组成

其元素分别为: a[0], a[1], a[2], a[3]

其序号从0开始。若存放首地址为2000H, 则在内存中为:

2000H	2004H	2008H	200CH	2010H
a[0]	a[1]	a[2]	a[3]	

C++不允许对数组的大小作动态的定义, 即数组的大小不能是变量, 必须是常量。

如果要根据不同的数值改变数组的大小，可用常量表达式。如：

```
#define SIZE 50
```

```
void main(void)
```

```
{ int art[SIZE];
```

```
.....
```

```
}
```

二、一维数组元素的引用

数组必须先定义，具体引用时（赋值、运算、输出）其元素等同于变量。

```
void main(void )
```

```
{  int i, a[10];
```

定义

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

赋值

```
        a[i]=i;
```

```
    for ( i=9; i>=0 ; i--)
```

```
        cout<<a[i]<<'\t';
```

```
    cout<<"\n";
```

输出

```
}
```

i=0, a[0]=0

i=1, a[1]=1

i=2, a[2]=2

i=9, a[9]=9

a

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

输出： 9 _ 8 _ 7 _ 6 _ 5 _ 4 _ 3 _ 2 _ 1 _ 0

三、一维数组的初始化

在定义数组的同时给数组元素赋值。

注意：

1、对数组中的一部分元素列举初值，未赋值的部分是0。

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

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

2、不能给数组整体赋值，只能一个一个地赋值。

```
int a[10]= {0,1,2,.....,9};
```

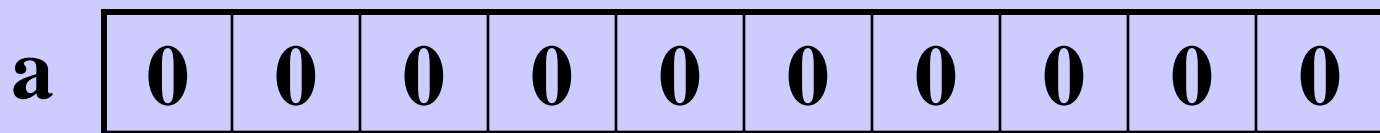
非法

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

3、可以用 `int a[] = {0,1,2,3,4,5,6,7,8,9};` 给数组赋值，编译器会自动计算出内的元素项数，并将数组定义为该长度。

4、用局部static 或全局定义的数组不赋初值，系统均默认其为 `'\0'`。

`static int a[10];`（即存储在静态数据区中的数组其元素默认为0）



数组在内存中顺序存放，第一个元素位于地址的最低端。

求Fibonacci数列：1,1,2,3,5,8,.....的前20个数，即

$$F_1=1 \quad (n=1)$$

$$F_2=1 \quad (n=2)$$

$$F_n=F_{n-1}+F_{n-2} \quad (n \geq 3)$$

f[0] f[1] f[2] f[3] f[4] f[5] f[5] f[6] f[7]

1	1	2	3	5	8	13	21		
----------	----------	----------	----------	----------	----------	-----------	-----------	--	--

$$f[i]=f[i-1]+f[i-2]$$

```
void main (void)
```

```
{ int i;
```

```
int f [20]={1,1};
```

```
for (i=2 ; i<20 ; i++ )
```

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

```
for ( i=0; i<20; i++)
```

```
{ if (i%5==0) cout<<"\n";
```

```
    cout<<f [i]<<"\t";
```

```
}
```

```
}
```


下面程序的运行结果是：

```
void main(void)
```

```
{ int a[6], i;
```

```
  for (i=1; i<6; i++)
```

```
    { a[i]=9*(i-2+4*(i>3))%5 ;
```

```
      cout<<a[i]<<'\t';
```

```
    }
```

```
}
```

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

随机	-4	0	4	4	3
----	----	---	---	---	---

i	1	2	3	4	5
a[i]	-4	0	4	4	3

输出： -4 0 4 4 3

排序算法

用起泡法对6个数排序（由小到大）

将相邻的两个数两两比较，将小的调到前头。

9	8	8	8	8	8	8	5	5	5	5	5	4	4	4
8	9	5	5	5	5	5	8	4	4	4	4	5	3	3
5	5	9	4	4	4	4	4	8	2	3	3	3	5	0
4	4	4	9	2	2	2	2	2	8	0	0	0	0	5
2	2	2	2	9	0	0	0	0	0	8	8	8	8	8
0	0	0	0	0	9	9	9	9	9	9	9	9	9	9

第一趟

循环5次

第二趟

循环4次

第三趟

循环3次

4 3 3

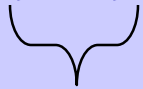
3 4 0

0 0 4

5 5 5

8 8 8

9 9 9



第四趟

循环2次

3 0

0 3

4 4

5 5

8 8

9 9

第五趟

循环1次

总结:

	共有6个数					n
趟数	1	2	3	4	5	$j(1 \sim n-1)$
次数	5	4	3	2	1	$n-j$

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

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

```
        { if (a[i]>a[i+1])
```

```
            {  $t=a[i]$ ;
```

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

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

```
        }
```

```
    }
```

一般，元素的序号从0开始，因此，程序可以变动如下：

```
for (j=0; j<n-1; j++)
```

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

```
        { if (a[i]>a[i+1])
```

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

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

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

```
            }
```

```
        }
```

二维数组的定义和引用

一、定义方式:

类型说明符 数组名[常量表达式][常量表达式];

The diagram shows the code `int a[3][4];` with four callout boxes:
- A red box labeled '定义类型' (Define type) pointing to `int`.
- A red box labeled '数组名' (Array name) pointing to `a`.
- A red box labeled '行数' (Number of rows) pointing to the first dimension `3`.
- A red box labeled '列数' (Number of columns) pointing to the second dimension `4`.

表明a数组由 3×4 个int型元素组成

其元素分别为: `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]`

其元素分别为： $a[0][0]$, $a[0][1]$, $a[0][2]$, $a[0][3]$,
 $a[1][0]$, $a[1][1]$, $a[1][2]$, $a[1][3]$,
 $a[2][0]$, $a[2][1]$, $a[2][2]$, $a[2][3]$

其行列的序号均从0开始。若存放首地址为2000H，
则在内存中为：

2000H		2008H		2010H		2014H		201cH		2020H		2028H		202cH	
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]				

即在内存中，多维数组依然是直线顺序排列的，第一个元素位于最低地址处。

二、二维数组的引用

与一维数组一样，二维数组必须先定义，其维数必须是常量。具体引用时（赋值、运算、输出）其元素等同于变量。

输入：1 2 3 4 5 6<CR>

输出： 1 2 3
 4 5 6

```
void main(void)
```

定义

```
{ int a[2][3], i, j;
```

```
    cout<<"Input 2*3 numbers\n";
```

```
    for (i=0; i<2; i++) /* 输入 */
```

```
        for(j=0; j<3; j++)
```

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

赋值

```
    for (i=0; i<2; i++) /* 输出 */
```

```
        { for(j=0; j<3; j++)
```

```
            cout<<a[i][j]<<"\t";
```

```
            cout<<"\n";
```

输出

```
        }
```

```
    }
```

三、二维数组的初始化

在定义数组的同时给数组元素赋值。即在编译阶段给数组所在的内存赋值。

1、分行赋值

```
int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
```

2、顺序赋值

```
int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12}; //依次赋值
```


3、部分赋值

```
int a[3][4]={1,5,9};
```

/ a[0][0]=1, a[1][0]=5, a[2][0]=9 其余元素为0 */*

1	0	0	0
5	0	0	0
9	0	0	0

0	1	0	0
5	0	0	0
0	0	0	0

```
int a[3][4]={0,1,5};
```

/ a[0][0]=0, a[0][1]=1, a[1][0]=5 */*

4、分行或全部赋值时，可以省略第一维，第二维不可省。

```
int a[ ][4]={1,2},{5,6,7,8},{9,10,11,12};
```

5、不能给数组整体赋值，只能一个一个地赋值。

```
int a[2][3]={1,2,3,.....,12};
```

6、用static 定义的数组不赋初值，系统均默认其为‘\0’。

```
static int a[2][3];
```

```
void main(void)
```

```
{  int  a[3][3], i, j;  
    for (i=0; i<3; i++)  
    { for (j=0; j<3; j++)
```

```
        if (i==2)
```

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

```
        else
```

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

```
        cout<<a[i][j]<<'\t';
```

```
    }
```

```
    cout<<"\n";
```

```
}
```

i=0 a[0][0]=0 a[0][1]=1 a[0][2]=2

i=1 a[1][0]=0 a[1][1]=1 a[1][2]=2

a[2][0]=a[1][a[1][0]]+1=a[1][0]+1=1

a[2][1]=a[1][a[1][1]]+1=a[1][1]+1=2

a[2][2]=a[1][a[1][2]]+1=a[1][2]+1=3

输出: __0__1__2

 __0__1__2

 __1__2__3

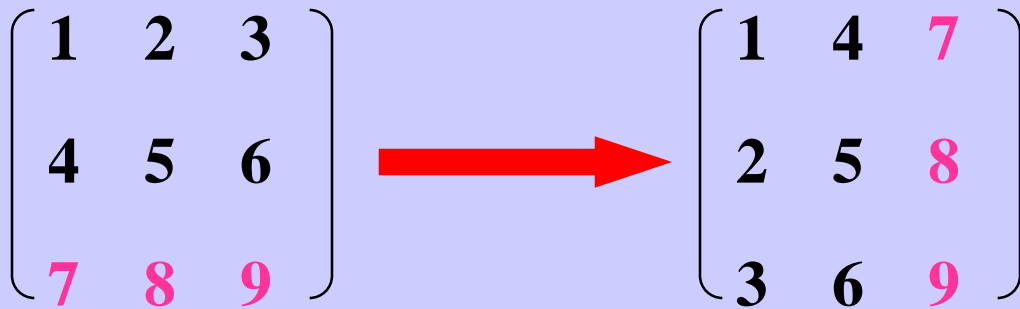
有一个 3×4 的矩阵，要求编程序求出其中值最大的那个元素的值，以及其所在的行号和列号。

先考虑解此问题的思路。从若干个数中求最大者的方法很多，我们现在采用“打擂台”算法。如果有若干人比武，先有一人站在台上，再上去一人与其交手，败者下台，胜者留台上。第三个人再上台与在台上者比，同样是败者下台，胜者留台上。如此比下去直到所有人都上台比过为止。最后留在台上的就是胜者。

程序模拟这个方法，开始时把a[0][0]的值赋给变量max，**max**就是开始时的擂主，然后让下一个元素与它比较，将二者中值大者保存在max中，然后再让下一个元素与新的max比，直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。

```
max=a[0][0]; //使max开始时取a[0][0]的值  
for (i=0;i<=2;i++) //从第0行到第2行  
    for (j=0;j<=3;j++) //从第0列到第3列  
        if (a[i][j]>max)//如果某元素大于max  
        {  
            max=a[i][j]; //max将取该元素的值  
            row=i; //记下该元素的行号i  
            colum=j; //记下该元素的列号j  
        }  
cout<<row<<'\t'<<colum<<'\t'<<max<<endl;
```

将数组行列式互换。



```
for (i=0; i<3; i++)
```

```
    for (j=0; j<3; j++)
```

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

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

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

```
        }
```

```
for (i=0; i<3; i++)
```

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

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

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

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

```
        }
```

打印杨辉三角形

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

$$a[i][j]=a[i-1][j-1]+a[i-1][j]$$


```
void main(void)
```

```
{    static int n[2],i,j,k;
```

```
    for(i=0;i<2;i++)
```

```
        n[j++]=n[i]+i+1;
```

```
    cout<<n[k]<<'\\t'<<n[k++]<<endl;
```

```
}
```

2 1

以下程序用于从键盘上输入若干个学生的成绩，统计出平均成绩，并输出低于平均成绩的学生成绩。输入负数结束

```
void main()
```

```
{ float x[100],sum=0, ave,a;
```

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

```
cout<<"Input score\n";
```

```
cin>>a;
```

```
while(a>=0)
```

```
{ x[n]=a;
```

```
sum+=a;
```

```
n++
```

```
cin>>a;
```

```
}
```

```
ave=sum/n;
```

```
cout<<"ave="<<ave<<endl;
```

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

```
if(x[i]<ave)
```

```
cout<<"x["<<i<<"]"<<x[i]<<endl;
```

输入一个十进制数，输出它对应的八进制数。

不断地除8，求其
余数，直到被除
数为0，最后余数
倒序排列。

$$725/2=362$$

$$362/2=181$$

$$181/2=90$$

$$90/2=45$$

$$45/2=22$$

$$22/2=11$$

$$11/2=5$$

$$5/2=2$$

$$2/2=1$$

$$1/2=0$$

$$\text{余数}=1=K_0$$

$$\text{余数}=0=K_1$$

$$\text{余数}=1=K_2$$

$$\text{余数}=0=K_3$$

$$\text{余数}=1=K_4$$

$$\text{余数}=0=K_5$$

$$\text{余数}=1=K_6$$

$$\text{余数}=1=K_7$$

$$\text{余数}=0=K_8$$

$$\text{余数}=1=K_9$$



```
void main(void)
```

```
{   int x , i, n ;
```

```
    int a[100];
```

```
    cin>>x;
```

```
    i=0;
```

```
    while(x)
```

```
    {
```

```
        a[i]=x%8;
```

```
        x=x/8;
```

```
        i++;
```

```
    }
```

```
    n=i;
```

```
    for(i=n-1;i>=0;i--)
```

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

```
    cout<<endl;
```

```
}
```



将余数依次
存入数组中

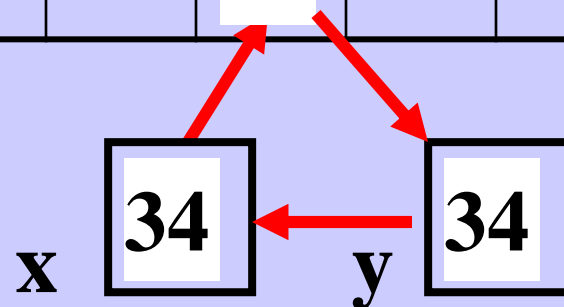
已有一个已排好序的数组, 今输入一个数, 要求按原来排序的规律将它插入数组中。

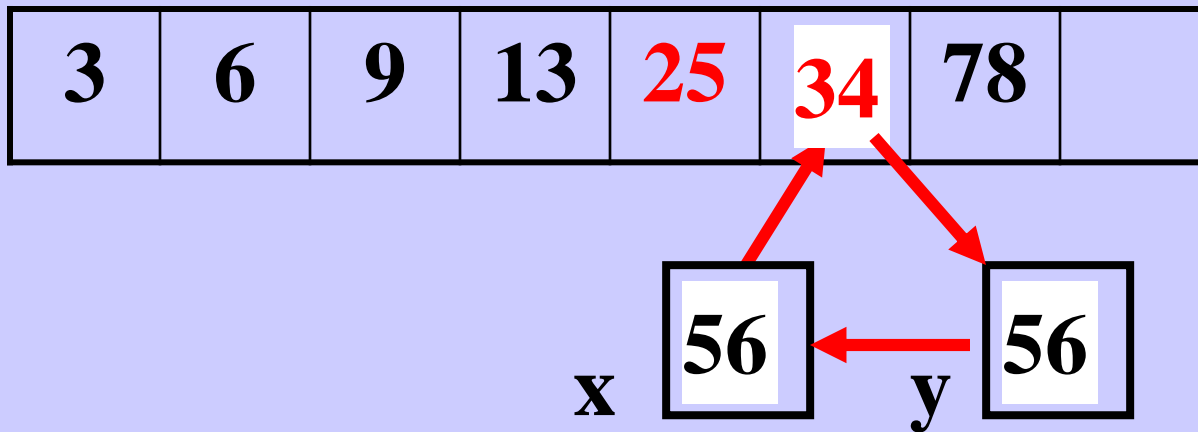
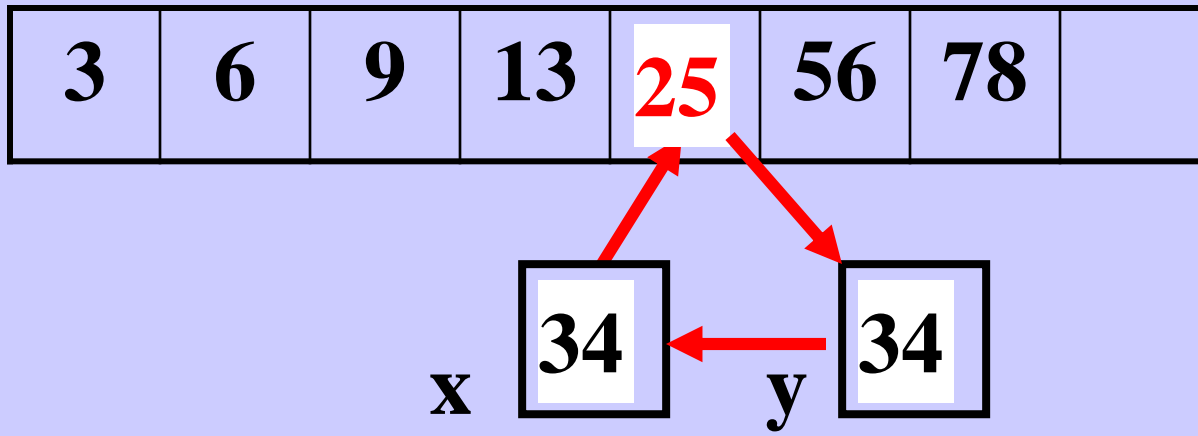
3	6	9	13	34	56	78	
---	---	---	----	----	----	----	--

输入: `cin>>x;` 25

3	6	9	13	25	34	56	78
---	---	---	----	----	----	----	----

3	6	9	13	25	56	78	
---	---	---	----	----	----	----	--





```

void main(void)
{ int a[6]={1,4,7,10,12};
  int x;
  for(int i=0;i<5;i++)
    cout<<a[i]<<"\t";
  cout<<endl;
  cout<<"Input x: ";
  cin>>x;
  for(i=0;i<5;i++)
  {  if(a[i]>x)
      break;
  }
}

```

将这个数插入

输入一个数

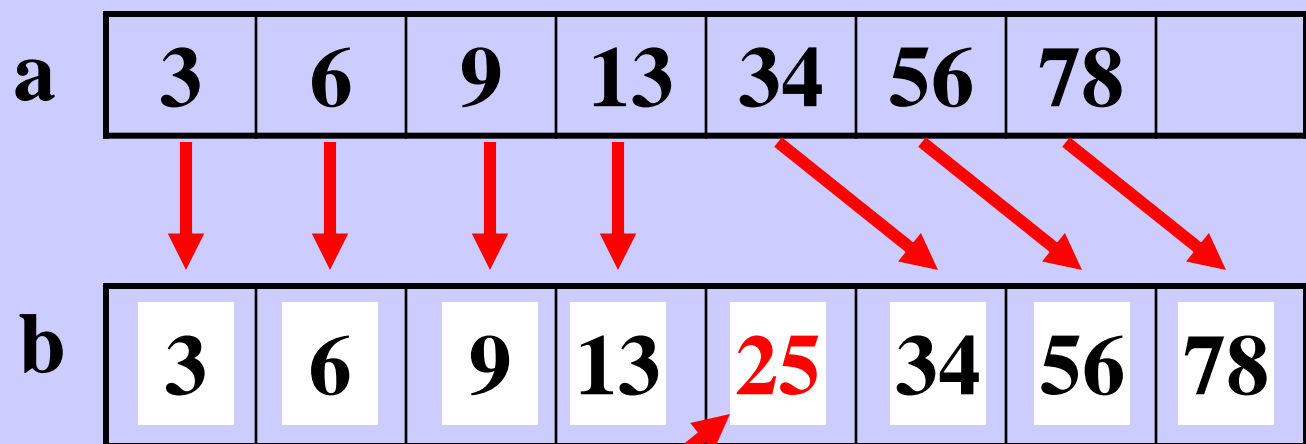
从头比较

大于这个数退出

```

for(int j=i;j<=5;j++)
{
    int y=a[j];
    a[j]=x;
    x=y;
}
for( i=0;i<6;i++)
    cout<<a[i]<<"\t";
cout<<endl;
}

```



输入: `cin>>x;` 25


```
void main(void)
```

```
{ int a[6]={1,4,7,10,12};
```

```
int b[6];
```

```
int x;
```

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

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

```
cout<<endl;
```

```
cout<<"Input x: ";
```

```
cin>>x;
```

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

```
    if(a[i]<x)
```

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

```
    else
```

```
        break;
```

重新开始赋值

```
b[i]=x;
```

```
for(int j=i;j<5;j++)
```

```
    b[j+1]=a[j];
```

```
for(i=0;i<6;i++)
```

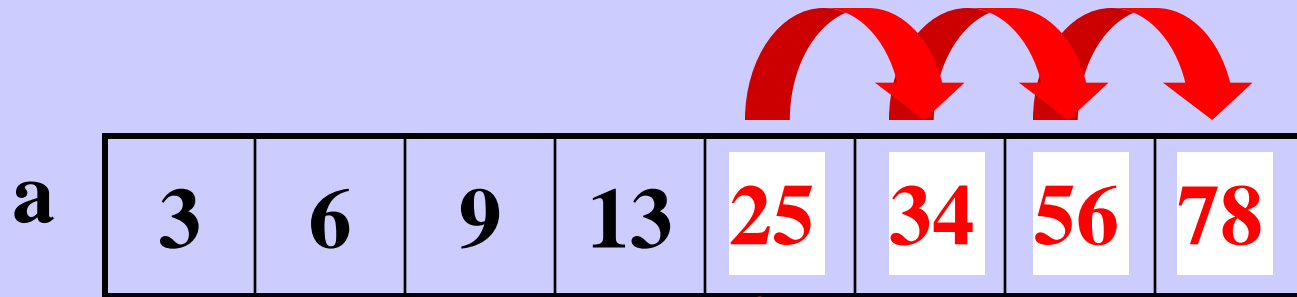
```
    cout<<b[i]<<"\t";
```

```
cout<<endl;
```

```
}
```

小于这个数赋值

大于这个数退出



输入: `cin>>x;` 25

`for(i=n-1;i>=0;i--)`

从后向前循环

```
void main(void)
{ int a[6]={2,5,8,10,12};
  int x;
  for(int i=0;i<5;i++)
      cout<<a[i]<<'\\t';
  cout<<endl;
  cout<<"Input x: ";
  cin>>x;
  for(i=4;i>0;i--)
  {   if(a[i]>x)
      a[i+1]=a[i];
      else
      break;
  }
```

a[i+1]=x;

赋值

for(i=0;i<6;i++)

cout<<a[i]<<'\\t';

cout<<endl;

}

关键！从后面开始循环

从前向后移数

不大于退出循环

用筛选法求出2~200之间的所有素数。

筛法：首先将1~n个数为数组置初值。2的倍数不是素数，置0； 3的倍数不是素数，置0； 5的倍数不是素数，置0；，依次类推，最后将数组中不是0的元素输出。

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

2	3	0	5	0	7	0	9	0	11	0	13	0	15	0	17	0	19	0
---	---	---	---	---	---	---	---	---	----	---	----	---	----	---	----	---	----	---

2	3	0	5	0	7	0	0	0	11	0	13	0	0	0	17	0	19	0
---	---	---	---	---	---	---	---	---	----	---	----	---	---	---	----	---	----	---

数组作为函数参数

一、数组元素作函数参数

数组元素作函数实参，用法与一般变量作实参相同，是“值传递”。

有两个数据系列分别为：

```
int a[8]={26,1007,956,705,574,371,416,517};
```

```
int b[8]={994,631,772,201,262,763,1000,781};
```

求第三个数据系列 c ， 要求c中的数据是a b中对应数的最大公约数。

```
int a[8]={26, 1007, 956, 705, 574, 371, 416, 517};
```

```
int b[8]={994, 631, 772, 201, 262, 763, 1000, 781};
```

```
    c[8]={2,  1,   4,   3,   2   , 7   , 8,   11}
```

```
int gys(int m,int n)
```

```
{ int r;
```

```
if(m<n) { r=m; m=n; n=r; }
```

```
while(r=m%n) { m=n; n=r; }
```

```
return n;
```

```
}
```

```
void main(void)
```

```
{ int a[8]={26,1007,956,705,574,371,416,517};
```

```
int b[8]={994,631,772,201,262,763,1000,781};
```

```
int c[8];
```

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

```
c[i]=gys(a[i],b[i]); //
```

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

```
cout<<c[i]<<"\t";
```

```
cout<<endl;
```

```
}
```

求m,n的最大公约数,
作为函数值返回

循环求对应数组元素
的最大公约数

二、用数组名作函数参数

在C++中，数组名被认为是**数组在内存中存放的首地址**。

用数组名作函数参数，**实参与形参都应用数组名**。

这时，**函数传递的是数组在内存中的地址**。

实参中的数组地址传到形参中，实参形参**共用同一段内存**。


```
void fun(int a[2])
```

```
{ for(int i=0;i<2;i++)
```

```
    a[i]=a[i]*a[i];
```

```
}
```

```
void main(void)
```

```
{ int b[2]={2,4};
```

```
    cout<<b[0]<<'\t'<<b[1]<<endl;
```

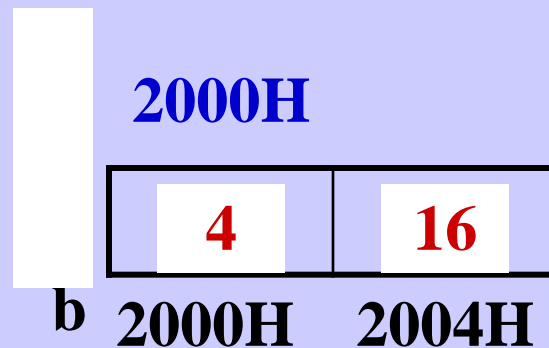
```
    fun(b);
```

```
    cout<<b[0]<<'\t'<<b[1]<<endl;
```

```
}
```

数组b和数组a占据同一段内存

a同样为数组首地址，也是2000H



b就是2000H

输出: 2 4

4 16

```
void sort(int x[ ], int n)
```

```
{ int t,i,j;
```

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

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

```
      if(x[j]>x[j+1])
```

```
        { t=x[j]; x[j]=x[j+1]; x[j+1]=t;}
```

```
    }
```

```
void main(void)
```

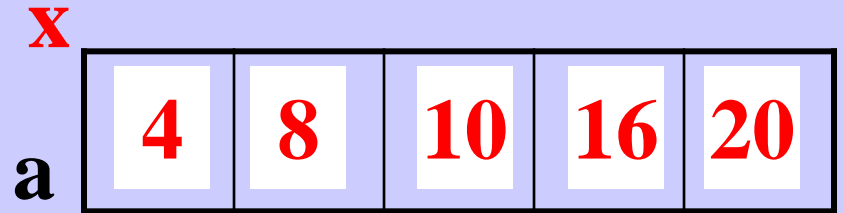
```
{ int a[5]={20,4,16,8,10};
```

```
  sort(a, 5 );
```

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

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

```
}
```



有一个一维数组，内放10个学生成绩，求平均成绩。

数组名作
函数形参

```
float average (float array[ ])
{ int i;
  float aver, sum=array[0];
  for (i=1; i<10; i++)
    sum=sum+array[i];
  aver=sum/10;
  return aver;
}
```

```
void main(void)
```

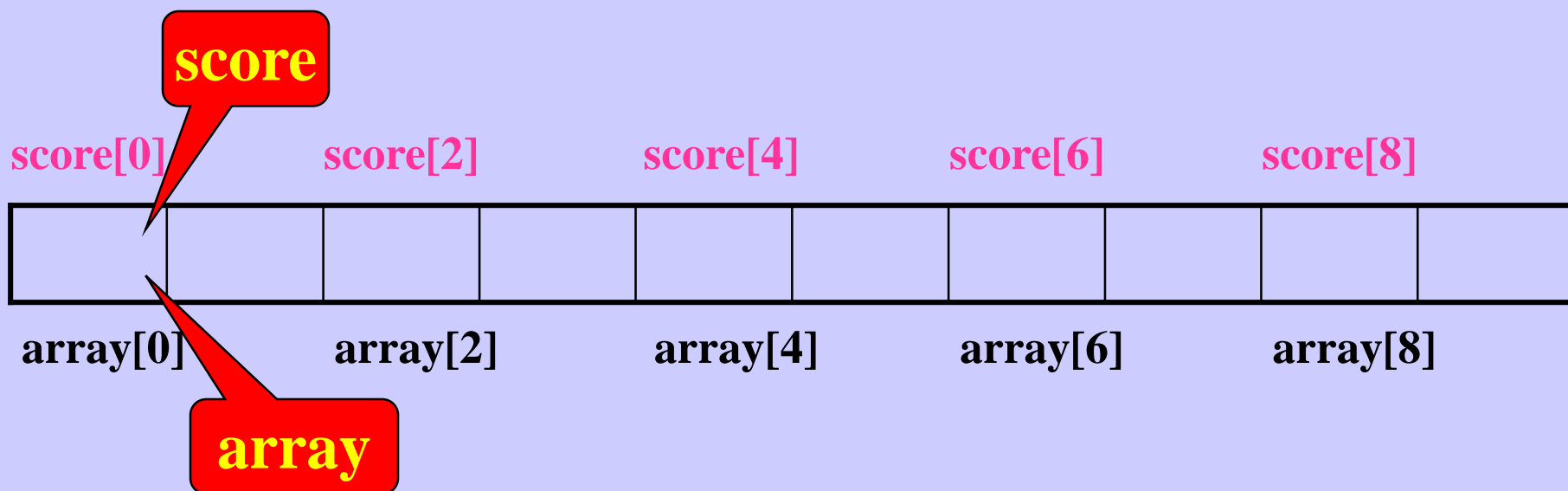
```
{ static float score[10]={ 100, 90, ...};
  float aver;
  aver=average(score);
  cout<<"aver="<<aver<<"\n";
}
```

数组名作
函数实参

注意：

- 1、用数组名作函数参数，应在主调函数和被调函数中分别定义数组，且类型一致。
- 2、需指定实参数组大小，形参数组的大小可不指定。数组名作实参实际上是传递数组的首地址。

3、C++语言规定，数组名代表数组在内存中存储的首地址，这样，数组名作函数实参，实际上传递的是数组在内存中的首地址。实参和形参共占一段内存单元，形参数组中的值发生变化，也相当于实参数组中的值发生变化。



程序中的函数p()用于计算:

主函数利用函数完成计算

$$p = \sum_{i=0}^{n-1} ax_i + by_i$$

$$s1 = \sum_{i=0}^7 d_i + 2v_i$$

$$s2 = \sum_{i=0}^9 3v_i + 4w_i$$

```
int d[]={2,3,5,4,9,10,8};
```

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

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

```
int p(int a, int x[], int b, int y[], int n)
```

```
{ int i, s;
```

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

```
    s+=a*x[i]+b*y[i];
```

```
  return s;
```

```
}
```

```
void main(void)
```

```
{cout<<"\ns1="<< p(1,d,2,v,8) ;
```

```
  cout<<"\ns2="<< p(3,v,4,w,10) ;
```

```
}
```

```
int a[10], i;
```

```
void sub1(void)
```

```
{ for(i=0;i<10;i++) a[i]=i+i; }
```

```
void sub2(void)
```

```
{ int a[10], max,i;
```

```
max=5; for(i=0;i<max;i++) a[i]=i;
```

```
}
```

```
void sub3(int a[])
```

```
{ int i;
```

```
for(i=0;i<10;i++) cout<<a[i]<<'\\t';
```

```
cout<<endl;
```

```
}
```

```
void main(void)
```

```
{ sub1();
```

```
sub3(a);
```

```
sub2();
```

```
sub3(a);
```

```
}
```

输出: 0 2 4 6 8 10 12 14 16 18

0 2 4 6 8 10 12 14 16 18

编写程序，在被调函数中删去一维数组中所有相同的数，使之只剩一个，数组中的数已按由小到大的顺序排列，被调函数返回删除后数组中数据的个数。

例如：

原数组： **2 2 2 3 4 4 5 6 6 6 6 7 7 8 9 9 10 10 10**

删除后： **2 3 4 5 6 7 8 9 10**

用多维数组名作函数参数

同样，实参向形参传递的是数组的首地址。

如果实参、形参是二维数组，则形参可以省略第一维，不可省略第二维，且第二维必须与实参中的维数相等。

`int score[5][10]`

`int array[][10]`

`int score[5][10]`

`int array[3][10]`

`int score[5][10]`

`int array[][8]`

错误

有一个 3×4 的矩阵，求其中的最大元素。

```
int max_value (int array[ ][4])
```

```
{ int i, j, k, max;
```

形参

```
max=array[0][0];
```

```
for (i=0; i<3; i++)
```

```
for (j=0; j<4; j++)
```

```
if (array[i][j]>max)
```

```
max=array[i][j];
```

```
return (max);
```

函数值

```
void main (void)
```

```
{ static int a[3][4]={ {1,3,5,7},
```

```
{2,4,6,8},{15,17,34,12}};
```

```
cout<<"max is "<<max_value(a)<<"\t";
```

```
}
```

实参

数组a与数组array共用一段内存

字符数组

用来存放字符数据的数组是字符数组，字符数组中的一个元素存放一个字符。

一、字符数组的定义

char 数组名[常量表达式];

类型

数组名

char c[4]; /*每个元素占一个字节*/

数组大小

c[0]='I'; c[1]='m'; c[2]='_';

二、字符数组的初始化

与数值数组的初始化相同，取其相应字符的**ASCII**值。

```
char c[10]={‘I’, ‘ ’, ‘a’, ‘m’, ‘ ’, ‘a’, ‘ ’, ‘b’, ‘o’, ‘y’};
```

	c[0]									c[9]	
c	‘I’	‘ ’	‘a’	‘m’	‘ ’	‘a’	‘ ’	‘b’	‘o’	‘y’	随机

```
char st1[ ]={65, 66, 68};
```

'A'	'B'	'D'
------------	------------	------------

如果字符个数大于数组长度，做错误处理；如果数值个数小于数组长度，后面的字节全部为 ‘\0’。

如果省略数组长度，则字符数即为数组长度。

```
static char c[ ]={'I', ' ', 'a', 'm', ' ', 'a', ' ', 'g', 'i', 'r', 'l'};
```

同理，也可定义和初始化一个二维或多维的字符数组。分层或省略最后一维。

三、字符数组的引用

```
void main(void)
```

```
{ char c[10]={‘I’, ‘ ’, ‘a’, ‘m’, ‘ ’, ‘a’, ‘ ’, ‘b’, ‘o’, ‘y’};
```

```
    int i;
```

定义

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

```
        cout<<c[i];
```

```
    cout<<“\n”;
```

输出

```
}
```

四、字符串和字符串结束标志

C++语言将字符串作为字符数组来处理。

字符串常量：“CHINA”，在机内被处理成一个无名的字符型一维数组。

C	H	I	N	A	'\0'
---	---	---	---	---	------

C++语言中约定用 ‘\0’作为字符串的结束标志，它占内存空间，但不计入串长度。有了结束标志

‘\0’后，程序往往**依据它判断字符串是否结束**，而不是根据定义时设定的长度。

字符串与字符数组的区别:

```
char a[]={'C','H','I','N','A'};
```

字符数组

C	H	I	N	A	随机	随机
---	---	---	---	---	----	----

长度占5个字节

```
char c[]="CHINA";
```

字符串

C	H	I	N	A	'\0'	随机
---	---	---	---	---	------	----

长度占6个字节

可以用字符串的形式为字符数组赋初值

```
char c[ ]={"I am a boy"}; /*长度11字节, 以 '\0'结尾  
*/  
char a[ ]={'I', ' ', 'a', 'm', ' ', 'a', ' ', 'b', 'o', 'y'};  
/* 长度10字节 */
```

如果数组定义的长度大于字符串的长度, 后面均为 '\0'。

```
char c[10]="CHINA";
```

c	C	H	I	N	A	\0	\0	\0	\0	\0
---	---	---	---	---	---	----	----	----	----	----

'\0'的ASCII为0, 而 ' ' (空格)的ASCII为32。

```
char w[ ]={'T', 'u', 'r', 'b', 'o', '\0'};
```

```
char w[ ]={"Turbo\0"};
```

```
char w[ ]="Turbo\0";
```

```
char w[ ]='Turbo\0';
```

非法

T	u	r	b	o	'\0'
T	u	r	b	o	'\0'
T	u	r	b	o	'\0'

```
char a[2][5]={“abcd”, “ABCD”};
```

a	b	c	d	‘\0’
A	B	C	D	‘\0’

在语句中字符数组不能用赋值语句整体赋值。

```
char str[12];          char str[12]=“The String”;
```

```
str=“The String”;
```

非法，在语句中赋值


定义数组，开辟空间时赋初值

str为字符数组在内存中存储的地址，一经定义，便成为常量，不可再赋值。

字符数组的输入输出

逐个字符的输入输出。这种输入输出的方法，通常是使用循环语句来实现的。如：

```
char str[10];
```



```
cout<<“输入十个字符： ”；
```

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

```
cin>>str[i];
```



```
//A
```

```
.....
```

A行将输入的十个字符依次送给数组str中的各个元素。

把字符数组作为字符串输入输出。对于一维字符数组的输入，在cin中仅给出**数组名**；输出时，在cout中也只给出**数组名**。

```
void main (void )
```

输入： abcd<CR>

```
{char s1[50],s2[60];
```

string<CR>

```
cout << “输入二个字符串:”;
```

```
cin >> s1;
```

数组名

```
cin >> s2;
```

数组名

```
cout << “\n s1 = “ << s1;
```

```
cout << “\n s2 = “ << s2 << “\n”;
```

```
}
```

输出到 ‘\0’为止

cin只能输入一个单词，不能输入一行单词。

当要把**输入的一行**作为一个字符串送到字符数组中时，则要使用函数**cin.getline()**。这个函数的第一个参数为字符数组名，第二个参数为允许输入的最大字符个数。

cin.getline(数组名, 数组空间数);

首先开辟空间

char s1[80];

.....

参数是数组名

cin.getline(s1, 80);

```
void main (void )
```

```
{  char s3[81];
```

定义

```
    cout<<"输入一行字符串:";
```

```
    cin.getline(s3,80);
```

从键盘接收一行字符

```
    cout<<"s3="<<s3<<' \n' ;           //B
```

```
}
```

输出到 '\0' 为止

当输入行中的字符个数小于80时，将实际输入的字符串（不包括换行符）全部送给s3；当输入行中的字符个数大于80时，只取前面的80个字符送给字符串。

从键盘输入一行字符，统计其中分别有多少大小写字母，以\$号结束输入。

从键盘输入一行字符，统计其中分别有多少大小写字母。

从键盘输入一行字符，其中的大写变小写，小写变大写。

从键盘接收一行字符，统计有多少个单词数？

we are students.

	w	e			a	r	e		s
	字母	字母	空格	空格	字母	字母	字母	空格	字母
0	1	1	0	0	1	1	1	0	1

不能用字母数或空格数来判断，只能用字母和空格**状态变化**的次数来判断。

设状态变量word，判别到字母时word为1，判别到非字母时word为0。

word的初始值为0，当从0变为1时，单词数加1。

```
void main(void)
{char s[80];
int i=0, word=0,num=0;
cin.getline (s,80);
while(s[i]!='\0')
```

表明前一字符非字母

```
{ if((s[i]>='a'&& s[i]<='z' || s[i]>='A'&& s[i]<='Z')&& word==0)
```

```
{   word=1;
    num++;
}
```

改变状态，防止继续对
下一字母计数

```
else if(s[i]==' ' || s[i]=='\t')
```

```
    word=0;
```

改变状态，碰到下一个
字母时开始计数

```
    i++;
```

```
}
```

```
cout<<"num="<<num<<endl;
```

```
}
```

六、字符串处理函数

C++中没有对字符串变量进行赋值、合并、比较的运算符，但提供了许多字符串处理函数，用户可以调用 `#include "string.h"`

所有字符串处理函数的实参都是字符串数组名

1、合并两个字符串的函数 `strcat (str1, str2)`

空间足够大

```
static char str1[20]={“I am a ”};
```

```
static char str2[ ]={“boy”};
```

```
strcat (str1, str2);
```

I		a	m		a		'\0'	'\0'					
---	--	---	---	--	---	--	------	------	--	--	--	--	--

b	o	y	'\0'
---	---	---	------

I		a	m		a		b	o	y	'\0'			
---	--	---	---	--	---	--	---	---	---	------	--	--	--

将第二个字符串 `str2` 接到第一个字符串 `str1` 后。

注意：第一个字符串要有足够的空间。

2、复制两个字符串的函数 strcpy (str1, str2)

```
static char str1[20]={“I am a ”};
```

```
static char str2[ ]={“boy”};
```

```
strcpy (str1, str2);
```

str1

I		a	m		a		\0	\0					
---	--	---	---	--	---	--	----	----	--	--	--	--	--

str2

b	o	y	\0
---	---	---	----

str1

b	o	y	\0		a		\0	\0					
---	---	---	----	--	---	--	----	----	--	--	--	--	--

```
strcpy ( str1, “CHINA”);
```

字符串正确赋值

str1

C	H	I	N	A	\0								
---	---	---	---	---	----	--	--	--	--	--	--	--	--

```
str1=str2; str1=“CHINA”;
```

均为非法

```
strcpy (“CHINA”, str1);
```

3、比较两个字符串的函数 `strcmp (str1, str2)`

此函数用来比较str1和str2中字符串的内容。函数对字符串中的ASCII字符**逐个两两比较**，直到遇到不同字符或 ‘\0’ 为止。函数值由两个对应字符相减而得。

该函数具有返回值，返回值是两字符串对应的**第一个不同**的ASCII码的差值。

若两个字符串完全相同，函数值为0。

```
if ( strcmp (str1, str2)==0)
```

```
{ ..... }
```

用来判断两字符串是否相等

```
static char str1[20]={“CHINA”};
```

```
static char str2[ ]={“CHINB”};
```

```
cout<< strcmp (str1, str2)<<endl;
```

输出: -1

```
static char str1[20]={“CHINA”};
```

```
static char str2[ ]={“AHINB”};
```

```
cout<<strcmp (str1, str2)<<endl;
```

输出: 2

```
if (str1==str2) cout<<“yes\n”;
```

非法

正确

```
if (strcmp (str1,str2)= =0)    cout<<“yes\n”;
```

4、求字符串长度的函数 `strlen (str1)`

函数参数为数组名，返回值为数组首字母到 ‘\0’ 的长度。

并非数组在内存中空间的大小。长度不包括 ‘\0’。

```
char s[80];
```

```
strcpy(s, "abcd");
```

```
cout<<strlen(s)<<endl;      输出： 4
```

```
cout<<sizeof(s)<<endl;      输出： 80
```

str1

b	o	y	\0		a		\0	\0					
---	---	---	----	--	---	--	----	----	--	--	--	--	--

```
cout<<strlen(str1)<<endl;    输出： 3
```



```
char str1[20]={“CHINA”};
```

```
cout<<strlen (str1)<<endl;
```

输出： 5

```
char str1[20]={“a book”};
```

```
cout<<strlen (str1)<<endl;
```

输出： 6

```
char sp[ ]={“\t\v\\0will\n”};
```

```
cout<<strlen (sp)<<endl;
```

输出： 3

```
char sp[ ]={“\x69\082”};
```

```
cout<<strlen (sp)<<endl;
```

输出： 1

5、 **strlwr (str1)**

将**str1**中的大写字母转换成小写字母。

6、 **strupr (str1)**

将**str1**中的小写字母转换成大写字母。

7、函数strncmp(字符串1,字符串2 , maxlen)

函数原型为:

int strncmp(char str1[], char str2[],int m)

第三个参数为正整数，它限制了至多比较的字符个数

若字符串1或字符串2的长度小于maxlen的值时，函数的功能与strcmp()相同。

当二个字符串的长度均大于maxlen的值时，maxlen为至多要比较的字符个数。

输出：0

```
cout<<strncmp("China","Chifjsl;kf",3)<<"\n";
```

8、函数strncpy(字符数组名1, 字符串2, maxlen)

函数原型为:

```
void strncpy(char str1[ ], char str2[ ],int m)
```

第三个参数为正整数，它限制了至多拷贝的字符个数

若字符串2的长度小于maxlen的值时，函数的功能与strcpy()相同。

当字符串2的长度大于maxlen的值时，maxlen为至多要拷贝的字符个数。

空间足够大

```
char s[90],s1[90];
```

```
strncpy(s,"abcdssfsdfk",3);           //A
```

```
strncpy(s1,"abcdef ",90);           //B
```

```
cout<<s<<endl;      输出: abc
```

```
cout<<s1<<endl;     输出: abcdef
```

注意，二字符串之间不能直接进行比较，赋值等操作，这些操作必须通过字符串函数来实现。

输入三个字符串按大小输出。

输入n个字符串按大小输出。

```
void changed(char str1[],char str2[])  
{char str3[80];  
  strcpy(str3,str1);  
  strcpy(str1,str2);  
  strcpy(str2,str3);  
}
```

```
void main(void)  
{char s1[80],s2[80],s3[80];  
  cout<<"Input 3 strings:\n";  
  cin.getline(s1);  
  cin.getline(s2);  
  cin.getline(s3);  
  if(strcmp(s1,s2)>0) changed(s1,s2);  
  if(strcmp(s1,s3)>0) changed(s1,s3);  
  if(strcmp(s2,s3)>0)changed(s2,s3);  
  cout<<"sorted:"<<endl<<endl;  
  cout<<s1<<endl<<s2<<endl<<s3<<endl;  
}
```

```
void changed(char str1[],char str2[])
{char str3[80];
 strcpy(str3,str1);
 strcpy(str1,str2);
 strcpy(str2,str3);
}
```

```
void main(void)
{char ss[10][80];
 int i,j;
 cout<<"Input 10 strings:\n";
 for(i=0;i<10;i++)
     cin.getline (ss[i],80);
 for(i=0;i<9;i++)
     for(j=0;j<9-i;j++)
         if(strcmp(ss[j],ss[j+1])>0)
             changed(ss[j],ss[j+1]);
 cout<<"sorted:"<<endl<<endl;
 for(i=0;i<10;i++)
     cout<<ss[i]<<endl;
}
```


用选择法对6个数排序（由小到大）

设定一个变量，放入数组中的最小数的序号，然后将其与最上面的数比较交换。

1、min=1 2、a[min]与a[2]比较 3、min=2 4、a[min]与a[3]比较

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

假定元素序号为1的数是最小的数

即9与8比较

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

这时，最小数的序号变为2

即8与5比较

min=3

a[min]与a[4]比较

min=4

a[min]与a[5]比较

min

3

这时，最小数的序号变为3

9 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

0 a[6]

即5与4比较

min

4

这时，最小数的序号变为4

9 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

0 a[6]

即4与2比较

min=5

a[min]与a[6]比较

min=6

a[min]与a[1]交换

min

5

这时，最小数的序号变为5

9 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

0 a[6]

第一趟比较完毕，
最小数是a[6]，最小数的序号为6

第一趟，循环5次

min

6

0 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

9 a[6]

min=2

a[min]与a[3]比较

min=3

a[min]与a[4]比较

min

2

0 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

9 a[6]

min

3

0 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

9 a[6]

从第二个数开始
比较，假定最小
数的序号为2

min=4

a[min]与a[5]比较

min=5

a[min]与a[6]比较

min

4

0 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

9 a[6]

min

5

0 a[1]

8 a[2]

5 a[3]

4 a[4]

2 a[5]

9 a[6]

83

min=5

a[min]与a[2]交换

min

5

0 a[1]

2 a[2]

5 a[3]

4 a[4]

8 a[5]

9 a[6]

第二趟比较完毕，
最小数是**a[5]**，最小
数的序号为**5**

第二趟，循环4次

min=3

a[min]与a[4]比较

min=4

a[min]与a[5]比较

min

3

0 a[1]

2 a[2]

5 a[3]

4 a[4]

8 a[5]

9 a[6]

min

4

0 a[1]

2 a[2]

5 a[3]

4 a[4]

8 a[5]

9 a[6]

min=4

a[min]与a[6]比较

min=4

a[min]与a[3]交换

min

4

0 a[1]

2 a[2]

5 a[3]

4 a[4]

8 a[5]

9 a[6]

min

4

0 a[1]

2 a[2]

4 a[3]

5 a[4]

8 a[5]

9 a[6]

第三趟，循环3次

min=4

a[min]与a[5]比较

min=4

a[min]与a[6]比较

min

4

0 a[1]

2 a[2]

4 a[3]

5 a[4]

8 a[5]

9 a[6]

min

4

0 a[1]

2 a[2]

4 a[3]

5 a[4]

8 a[5]

9 a[6]

min=4

a[min]与a[4]交换

min

4

0 a[1]

2 a[2]

4 a[3]

5 a[4]

8 a[5]

9 a[6]

第四趟，循环2次

min=5

a[min]与a[6]比较

min=5

a[min]与a[5]交换

min

5

0 a[1]

2 a[2]

4 a[3]

5 a[4]

8 a[5]

9 a[6]

min

5

0 a[1]

2 a[2]

4 a[3]

5 a[4]

8 a[5]

9 a[6]

第五趟，循环1次

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

{ min=i ;

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

if (a[min]>a[j]) min=j ;

t=a[min];

a[min]=a[i];

a[i]=t;

}

总结:

	共有6个数					n
趟数	1	2	3	4	5	i(1~n-1)
次数	5	4	3	2	1	n-i

一般，元素的序号从0开始，因此，程序可以变动如下：

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

```
{ min=i;
```

每一次循环前设置最小数的序号

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

```
if (a[min]>a[j])
```

```
min=j ;
```

小循环，找最小数的序号，从 i 找起

```
t=a[min];
```

```
a[min]=a[i];
```

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

大循环，找到后与 i 交换

```
}
```


调试程序的方法：

- 1) 单步调试：以行为单位，每运行一步，程序就会中断，可以实时查询目前各变量的状态及程序的走向。可以选择是否进入子函数。
- 2) 运行到光标处，可以直接使程序运行到光标处再进行单步调试，这种方法可以不必运行正确的循环而直接到有疑问的地方。

在a数组中查找与x值相同的元素所在的位置，数据从a[1]元素开始存放，请填空：

```
#define MAX 10
```

```
while(x!=    a[i]    )
```

```
void main(void)
```

```
    i--    ;
```

```
{ int a[MAX+1], x, i;
```

```
if(    i!=0    )
```

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

```
cout<<x<<"the pos:"<<i<<endl
```

```
cin>>    a[i]    ; else
```

```
cout<<"Enter x:";
```

```
cout<<"Not found"<<endl;
```

```
cin>>x;
```

```
a[0]=x; i=MAX;
```

```

void main(void)
{ char str[ ]="SSSWILTECH1\1\11W\1WALLMP1";
  char c; int k;
  for(k=2; (c=str[k])!='\0';k++)
  { switch(c)
    { case 'A' : cout<<'a'; continue;
      case '1': break;
      case 1:  while((c=str[++k])!='\1'&&c!='\0');
      case 9: cout<<'#';
      case 'E' :
                                     S W I T C H * # W a M P *
      case 'L': continue;
      default: cout<<c; continue;
    } cout<<'*';
  } cout<<endl;
}

```

以下程序分别在a数组和b数组中放入an+1和bn+1个由小到大的有序数，程序把两个数组中的数按由小到大的顺序归并到c数组中，请填空：

```
void main(void)
{  int a[10]={1,2,5,8,9,10},an=5;

    int b[10]={1,3,4,8,12,18}, bn=5;

    int i,j,k, c[20], max=9999;

    a[an+1]=b[an+1]=max;

    i=j=k=0;
```

```
    while( a[i]!=max||b[j]!=max)
    {  if(a[i]<b[j])
        {  c[k]=a[i];
            k++;
            i++
        }
        else
        {  c[k]=b[j];
            k++;
            j++;
        }
    }  for(i=0;i<k;i++)cout<<c[i];

    cout<<endl;

}
```

编写程序，在被调函数中删去一维数组中所有相同的数，使之只剩一个，数组中的数已按由小到大的顺序排列，被调函数返回删除后数组中数据的个数。

例如：

原数组： **2 2 2 3 4 4 5 6 6 6 6 7 7 8 9 9 10 10 10**

删除后： **2 3 4 5 6 7 8 9 10**