

第6章 利用数组处理批量数据

- 前几章使用的变量都属于基本类型，例如整型、字符型、浮点型数据，这些都是简单的数据类型。
- 对于有些数据，只用简单的数据类型是不够的，难以反映出数据的特点，也难以有效地进行处理。

- 如果有**1000**名学生，每个学生有一个成绩，要求这**1000**名学生的平均成绩。
- 用 **$s_1, s_2, s_3, \dots, s_{1000}$** 表示每个学生的成绩，**数组名**有**内在联系**。
- **C语言**用方括号中的数字表示下标，如用 **$s[15]$** 表示 不能用 **$s(15)$**

- 数组是一组有序数据的集合。数组中各数据的排列是有一定规律的，下标代表数据在数组中的序号
- 用一个数组名和下标唯一确定数组中的元素
- 数组中的每一个元素都属于同一个数据类型

6.1 怎样定义和引用一维数组

6.2 怎样定义和引用二维数组

6.3 字符数组

6.1 怎样定义和引用一维数组

6.1.1 怎样定义一维数组

6.1.2 怎样引用一维数组元素

6.1.3 一维数组的初始化

6.1.4 一维数组程序举例



6.1.1 怎样定义一维数组

- 一维数组是数组中最简单的
- 它的元素只需要用数组名加一个下标，就能唯一确定
- 要使用数组，必须在程序中先定义数组



6.1.1 怎样定义一维数组

➤ 定义一维数组的一般形式为：

类型符 数组名 [整型(常量)表达式];

➤ 数组名的命名规则和变量名相同

如 `int a[10];`

数组名



6.1.1 怎样定义一维数组

➤ 定义一维数组的一般形式为：

类型符 数组名[常量表达式];

➤ 数组名的命名规则和变量名相同

如 `int a[10];`

数组长度



6.1.1 怎样定义一维数组

- 定义一维数组的一般形式为：

每个元素的数据类型 表达式];

- 数组名的命名规则和变量名相同

如 **int** a[10];

10个元素：a[0], a[1], a[2], ..., a[9]

a[0]	a[1]	a[2]	a[3]	...	a[7]	a[8]	a[9]
------	------	------	------	-----	------	------	------



6.1.1 怎样定义一维数组

➤ 定义一维数组的一般形式为：

类型符 数组名[常量表达式];

int a[4+6]; 合法

int n=10;

int a[n];

不合法?

注意：P140 说明第3点。

旧版本标准不允许如此使用，

新版本标准可以动态定义数组。

但是不建议如此使用，存在兼容性、安全隐患。



6.1.1 怎样定义一维数组

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

a[0]	a[1]	a[2]	a[3]	...	a[7]	a[8]	a[9]
------	------	------	------	-----	------	------	------

特别注意：数组的下标从**0**开始

特别特别注意：数组越界问题

a[10] 就是越界，不是数组中的元素！



6.1.2 怎样引用一维数组元素

- 在定义数组并对其中各元素赋值后，就可以引用数组中的元素
- 注意：只能引用数组元素而不能一次整体调用整个数组全部元素的值



6.1.2 怎样引用一维数组元素

➤ 引用数组元素的表示形式为：

数组名 [下标]

如 $a[0]=a[5]+a[7]-a[2*3]$ 合法

int n=5,a[10];

a[n]=20;

合法



6.1.2 怎样引用一维数组元素

- 定义数组时的形式
- 引用数组元素的表示形式

int a[n]; 定义时, []中为整型常量

a[9]=10;

a[n]=20; 使用时, []下标为整型

int k[30];

k[30] = 100;

错误! 尤其注意数组越界的问题



➤ **double c[10];**

c[5.4]=23;

不合法

double shuzu[20],k=4;

shuzu[k]=9.0;

不合法?

6.1.2 怎样引用一维数组元素

例6.1 对**10**个数组元素依次赋值为**0,1,2,3,4,5,6,7,8,9**，要求按逆序输出。

➤ 解题思路：

- ◆ 定义一个长度为**10**的数组，数组定义为整型
- ◆ 要赋的值是从**0**到**9**，可以用循环来赋值
- ◆ 用循环按下标从大到小输出这**10**个元素



```
#include <stdio.h>
```

```
int main()
```

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

```
  for (i=0; i<=9;i++)
```

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

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

```
    printf("%d ",a[i]);
```

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

```
  return 0;
```

```
}
```

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

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

使a[0]~a[9]
的值为0~9



```
#include <stdio.h>
```

```
int main()
```

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

```
  for (i=0; i<=9;i++)
```

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

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

```
    printf("%d ",a[i]);
```

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

```
  return 0;
```

```
}
```

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

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

9 8 7 6 5 4 3 2 1 0

先输出a[9]，最后输出a[0]



6.1.3 一维数组的初始化

- 在定义数组的同时，给各数组元素赋值
- `int a[10]={0,1,2,3,4,5,6,7,8,9};`
- `int a[10]={0,1,2,3,4};` 相当于
`int a[10]={0,1,2,3,4,0,0,0,0,0};`
- `int a[10]={0,0,0,0,0,0,0,0,0,0};` 相当于
`int a[10]={0};`
- `int a[5]={1,2,3,4,5};` 可写为
`int a[]={1,2,3,4,5};` (不建议如此初始化)



➤ **int a[10]={1};** 表示什么含义?

```
int a[10]={1,0,0,0,0,0,0,0,0,0}
```

➤ 如果需要将数组中的**10**个元素都设为**1**
如何实现?

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

```
int i;
```

```
for(i=0;i<10;i++) a[i]=1;
```

6.1.4 一维数组程序举例

例6.2 用数组处理求Fibonacci数列问题

► 解题思路:

- ◆ 例5.8中用简单变量处理的，缺点不能在内存中保存这些数。假如想直接输出数列中第**25**个数，是很困难的。
- ◆ 如果用数组处理，每一个数组元素代表数列中的一个数，依次求出各数并存放在相应的数组元素中



```
#include <stdio.h>

int main()
{
    int i; int f[20]={1,1};
    for(i=2;i<20;i++)
        f[i]=f[i-2]+f[i-1];
    for(i=0;i<20;i++)
    {
        if(i%5==0) printf("\n");
        printf("%12d",f[i]);
    }
}
```

1	1	2	3	5
8	13	21	34	55
89	144	233	377	610
987	1597	2584	4181	6765



例6.3 有**10**个地区的面积，要求对它们按由小到大的顺序排列。

➤ 解题思路：

- ◆ 排序的规律有两种：一种是“升序”，从小到大；另一种是“降序”，从大到小
- ◆ 把题目抽象为：“对**n**个数按升序排序”
- ◆ 采用起泡法排序




```
for(i=0;i<5;i++)  
    if (a[i]>a[i+1])  
    { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

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

大数沉淀，小数起泡

第一趟比较，比较5次



```
for(i=0;i<4;i++)  
    if (a[i]>a[i+1])  
        { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

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

第二趟比较，比较4次



```
for(i=0;i<3;i++)  
    if (a[i]>a[i+1])  
    { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

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

第三趟比较，比较3次

```
for(i=0;i<2;i++)  
    if (a[i]>a[i+1])  
        { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

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

第四趟比较，比较2次



```
for(i=0;i<1;i++)  
    if (a[i]>a[i+1])  
    { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

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

第五趟比较，比较1次



```
for(i=0;i<5;i++)  
    if (a[i]>a[i+1])  
    { .....
```

```
for(i=0;i<4;i++)  
    if (a[i]>a[i+1])  
    { .....
```

.....

```
for(i=0;i<1;i++)  
    if (a[i]>a[i+1])  
    { .....
```

n个数，要进行n-1趟比较，第j趟比较，需两两比较n-j次

```
for(j=0;j<5;j++)  
    for(i=0;i<5-j;i++)  
        if (a[i]>a[i+1])  
        { .....
```



input 10 numbers :

34 67 90 43 124 87 65 99 132 26

int a[10];

printf("the sorted numbers :");

for (i=0;i<10;i++) printf("%d ",a[i]);

printf("\n");

for(j=0;j<9;j++)

for(i=0;i<9-j;i++)

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

{t=a[i];a[i]=a[i+1];a[i+1]=t;}

printf("the sorted numbers :\n");

for(i=0;i<10;i++) printf("%d ",a[i]);

printf("\n");

思考：若要从大到小排列，如何修改？



```
for(j=1;j<10;j++)  
  for(i=0;i<10-j;i++)  
    if (a[i]>a[i+1])  
    {  
      ↑ t=a[i];  
      a[i]=a[i+1];  
      ↓ a[i+1]=t;  
    }
```


6.2 怎样定义和引用二维数组

队员1 队员2 队员3 队员4 队员5 队员6

1分队	2456	1847	1243	1600	2346	2757
2分队	3045	2018	1725	2020	2458	1436
3分队	1427	1175	1046	1976	1477	2018

```
float pay[3][6];
```



6.2 怎样定义和引用二维数组

6.2.1 怎样定义二维数组

6.2.2 怎样引用二维数组的元素

6.2.3 二维数组的初始化

6.2.4 二维数组程序举例



6.2.1 怎样定义二维数组

- 二维数组定义的一般形式为

类型符 数组名[常量表达式][常量表达式];

如: **float a[3][4], b[5][10];**

- 二维数组可被看作是一种特殊的一维数组:

它的元素又是一个一维数组

- 例如, 把**a**看作是一个一维数组, 它有**3**个元素:

a[0]、a[1]、a[2]

- 每个元素又是一个包含**4**个元素的一维数组



a[0]	<u>a[0][0] a[0][1] a[0][2] a[0][3]</u>
a[1]	<u>a[1][0] a[1][1] a[1][2] a[1][3]</u>
a[2]	<u>a[2][0] a[2][1] a[2][2] a[2][3]</u>



逻辑存储

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] →

内存中的存储顺序



6.2.2 怎样引用二维数组的元素

➤ 二维数组元素的表示形式为：

数组名 [下标] [下标]

➤ $b[1][2]=a[2][3]/2$ 合法

➤ `for(i=0;i<m;i++)`

`printf("%d,%d\n",a[i][0],a[0][i]);` 合法



```
#include <stdio.h>

int main()
{
    int n=3,m=5,j=0,i=0;
    int a[n][m];
    for(j=0;j<=3;j++)
        for(i=0;i<=5;i++)
            a[j][i] = i*j ;

    .....

    .....
}
```



```
#include <stdio.h>
#define n 3;
#define m 5;
int main()
{  int j=0,i=0;
    int a[n][m];
    for(j=0;j<=3;j++)
        for(i=0;i<=5;i++)
            a[j][i] = i*j ;

    .....

    .....
}
```



6.2.3 二维数组的初始化

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

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

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

```
int a[3][4]={ {1,0,0,0},{5,0,0,0},  
              {9,0,0,0}};
```

```
int a[3][4]={ {1},{5,6}};相当于
```

```
int a[3][4]={ {1},{5,6},{0}};
```



6.2.3 二维数组的初始化

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

等价于:

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

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



6.2.3 二维数组的初始化

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

二维数组尤其注意下标越界的问题，比一维更容易出错

以上数组不存在 **a[3][4]** 元素，

下标最大的为 **a[2][3]**



6.2.4 二维数组程序举例

例**6.4** 将一个二维数组行和列的元素互换，存到另一个二维数组中。

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \longrightarrow b = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$



6.2.4 二维数组程序举例

➤ 解题思路:

- ◆ 可以定义两个数组：数组**a**为**2**行**3**列，存放指定的**6**个数
- ◆ 数组**b**为**3**行**2**列，开始时未赋值
- ◆ 将**a**数组中的元素**a[i][j]**存放到**b**数组中的**b[j][i]**元素中
- ◆ 用嵌套的**for**循环完成



```
#include <stdio.h>
```

```
int main()
```

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

```
int b[3][2],i,j;
```

```
printf("array a:\n");
```

```
for (i=0;i<=1;i++) 处理a的一行中各元素
```

```
{ for (j=0;j<=2;j++) 处理a中某一系列元素
```

```
{ printf("%5d",a[i][j]); 输出a的各元素
```

```
b[j][i]=a[i][j]; a元素值赋给b相应元素
```

```
}
```

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

```
}
```



```
printf("array b:\n");  
for (i=0;i<=2;i++)  
{ for(j=0;j<=1;j++)  
    printf("%5d",b[i][j]); 输出b的各元素  
    printf("\n");  
}  
return 0;  
}
```

```
array a:  
    1    2    3  
    4    5    6  
array b:  
    1    4  
    2    5  
    3    6
```



例6.5 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。

➤ 解题思路：采用“打擂台算法”

- ◆ 先找出任一人站在台上，第**2**人上去与之比武，胜者留在台上
- ◆ 第**3**人与台上的人比武，胜者留台上，败者下台
- ◆ 以后每一个人都与当时留在台上的人比武，直到所有人都上台比为止，最后留在台上的是冠军



例**6.5** 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。

➤ 解题思路：采用“打擂台算法”

◆ 先把 **$a[0][0]$** 的值赋给变量 **max**

◆ **max** 用来存放当前已知的最大值

◆ **$a[0][1]$** 与 **max** 比较，如果 **$a[0][1] > max$** ，则表示 **$a[0][1]$** 是已经比过的数据中值最大的，把它的值赋给 **max** ，取代了 **max** 的原值

◆ 以后依此处理，最后 **max** 就是最大的值



max=a[0][0]

for i=0 to 2

for j=0 to 3

真 **$a[i][j]>max$** **假**

max=a[i][j]

row=i

colum=j

输出: max,row,colum



```
max=10  
row=2  
column=1
```

.....

```
int i,j,row=0,column=0,max;  
int a[3][4]={ {1,2,3,4}, {9,8,7,6},  
              {-10,10,-5,2}};
```

max=a[0][0]; (不要遗漏, 改错)

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

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

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

```
            { max=a[i][j]; row=i; column=j; }
```

```
printf("max=%d\nrow=%d\n  
      column=%d\n",max,row,column);
```

.....

记最大值

行号

记列号



求最值的解题思路：

- ◆ 先把第一个元素的值赋给变量**max**（或**min**）
（改错）
- ◆ **max**（或**min**）用来存放当前已知的最大值
- ◆ **a[0][1]**与**max**比较，如果**a[0][1]>max**，
或**a[0][1]<min**， 把它的值赋给**max(min)**
取代了**max(min)**的原值
- ◆ 以后依此处理，最后**max(min)**就是最大（小）
的值



```
for(i=0; i<9; i++)  
    if (a[i]>a[i+1])  
    {  
        t=a[i];  
        a[i]=a[i+1];  
        a[i+1]=t;  
    }
```

冒泡法，单次循环
最大的元素为a[9]

6.3 字符数组

6.3.1 怎样定义字符数组

6.3.2 字符数组的初始化

6.3.3 怎样引用字符数组中的元素

6.3.4 字符串和字符串结束标志

6.3.5 字符数组的输入输出

6.3.6 善于使用字符串处理函数

6.3.7 字符数组应用举例



6.3.1 怎样定义字符数组

- 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
- 定义字符数组的方法与定义数值型数组的方法类似



6.3.1 怎样定义字符数组

```
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';
```

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

I		a	m		h	a	p	p	y
---	--	---	---	--	---	---	---	---	---



6.3.2 字符数组的初始化

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

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

I		a	m		h	a	p	p	y
---	--	---	---	--	---	---	---	---	---

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

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

c		p	r	o	g	r	a	m	\0
---	--	---	---	---	---	---	---	---	----

8进制的0 =0

6.3.2 字符数组的初始化

若提供的初值的个数大于数组长度，则语法错误。

如：

```
char c[6]={'c',' ','p','r','o','g','r','a','m'};
```

如果一个数组先后存放不同的字符串，数组长度应该大于最大的长度。

因此，编程人员在定义字符数组时，往往将数组的长度定义为较大一些，以保证足够的存储空间。

6.3.2 字符数组的初始化

```
char diamond[5][5]={{' ' , '/' , '/' , '*' },  
    {' ' , '/' , '*' , '/' , '/' },  
    {'*' , '/' , '/' , '/' , '/' },  
    {' ' , '/' , '*' , '/' , '*' },  
    {' ' , '/' , '/' , '*' , '/' } };
```



6.3.3怎样引用字符数组中的元素

例**6.6** 输出一个已知的字符串。

➤ 解题思路：

- ◆ 定义一个字符数组，并用“初始化列表”对其赋以初值
- ◆ 用循环逐个输出此字符数组中的字符



6.3.3 怎样引用字符数组中的元素

```
#include <stdio.h>
```

```
int main()
```

```
{ char c[15]={'I',' ','a','m',' ','a',  
            ' ','s','t','u','d','e','n','t','.'};
```

```
    int i;
```

```
    for(i=0;i<15;i++)
```

```
        printf("%c",c[i]);
```

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

```
    return 0;
```

```
}
```

```
I am a student.
```



6.3.3怎样引用字符数组中的元素

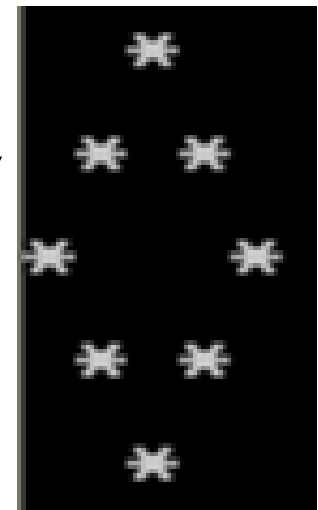
例**6.7** 输出一个菱形图。

➤解题思路：

- ◆定义一个字符型的二维数组，用“初始化列表”进行初始化
- ◆用嵌套的**for**循环输出字符数组中的所有元素。



```
#include <stdio.h>
int main()
{ char diamond[][5]={{' ',' ',' ',' ','*'}, {' '
  ,'*',' ',' ','*'},{'*',' ',' ',' ','*'},
  {' ',' ','*',' ','*'},{' ',' ',' ',' ','*'}};
  int i,j;
  for (i=0;i<5;i++)
    {for (j=0;j<5;j++)
      printf("%c",diamond[i][j]);
      printf("\n");
    }
  return 0;
}
```



6.3.4 字符串和字符串结束标志

- 在C语言中，是将字符串作为字符数组来处理的
- 关心的是字符串的有效长度而不是字符数组的长度
- 为了测定字符串的实际长度，C语言规定了字符串结束标志 **'\0'**

6.3.4 字符串和字符串结束标志

- **'\0'**代表**ASCII**码为**0**的字符
- 从**ASCII**码表可以查到，**ASCII**码为**0**的字符不是一个可以显示的字符，而是一个“空操作符”，即它什么也不做
- 用它作为字符串结束标志不会产生附加的操作或增加有效字符，只起一个供辨别的标志



6.3.4 字符串和字符串结束标志

```
char c[]={ "I am happy" };
```

可写成

```
char c[]="I am happy";
```

相当于

```
char c[11]={ "I am happy" };
```



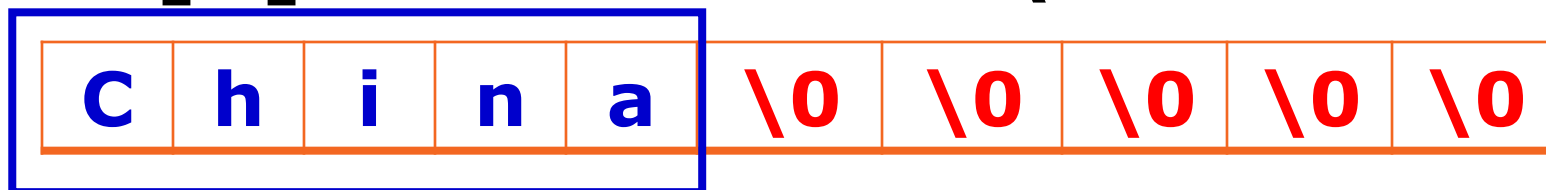
6.3.4 字符串和字符串结束标志

```
char c[10]={ "China" };
```

可写成

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

从c[5]开始，元素值均为\0



只显示

```
printf("%s",c);
```



6.3.4 字符串和字符串结束标志

系统会在字符串常量后自动添加‘\0’

字符串常量:用双引号引起来的字符串

如: “China” “hello world!\n”

“0” “ ” 等

char c[5] = “China”; 错误, 空间不够

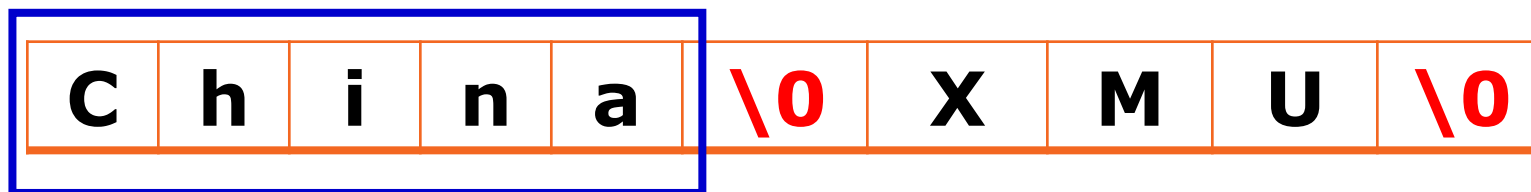


字符数组和字符串的区别

字符串一定以'\0' 结束;字符数组没有此要求

为了便于统一处理,一般都人为在字符数组后添加'\0' 结束

字符串'\0' 结束,之后的字符不做处理,如



```
printf("%s",c);
```



6.3.5 字符数组的输入输出

- 字符数组的输入输出可以有两种方法：
 - ◆ 逐个字符输入输出 (**%c**)
 - ◆ 整个字符串一次输入输出 (**%s**)
- 输出的字符中不包括结束符'**\0**'
- 用**%s**输出字符串时，**printf**函数中的输出项是字符数组名，不是数组元素名



6.3.5 字符数组的输入输出

- 如果一个字符数组中包含多个'\0'，则遇第一个'\0'时输出就结束

C	h	i	n	a	\0	X	M	U	\0
---	---	---	---	---	----	---	---	---	----

- 可以用**scanf**函数输入一个字符串
- **scanf**函数中的输入项**c**是已定义的字符数组名，输入的字符串应短于已定义的字符数组的长度



6.3.5 字符数组的输入输出

```
char c[6];
```

```
scanf("%s",c); China ✓
```

系统自动在**China**后面加一个'\0'

```
scanf("%s",c[0]); 错误!
```

记住: **%s** 对应数组名

```
scanf("%c",&c[0]); 对
```



6.3.5 字符数组的输入输出

```
char str1[5],str2[5],str3[5];  
scanf("%s%s%s",str1,str2,str3);
```

How are you? ✓

str1	H	o	w	\0	\0
------	---	---	---	----	----

str2	a	r	e	\0	\0
------	---	---	---	----	----

str3	y	o	u	?	\0
------	---	---	---	---	----



请特别注意: **scanf** 不接受带空格的字符串

```
char str1[5],str2[5],str3[5];  
scanf("%s%s%s",str1,str2,str3);
```

How are you? ✓

str1	H	o	w	\0	\0
str2	a	r	e	\0	\0
str3	y	o	u	?	\0



6.3.6 善于使用字符串处理函数

- 在C函数库中提供了一些用来专门处理字符串的函数，使用方便



6.3.6 善于使用字符串处理函数

1.puts函数----输出字符串的函数

➤ 其一般形式为:

puts (字符数组)

➤ 作用是将一个字符串输出到终端

```
char str[20]="China";
```

```
puts(str);
```

输出China



6.3.6 善于使用字符串处理函数

2. **gets**函数-----输入字符串的函数

➤ 其一般形式为:

gets(字符数组)

➤ 作用是输入一个字符串到字符数组

```
char str[20];
```

```
gets(str);
```

```
Computer ✓
```



6.3.6 善于使用字符串处理函数

2. **gets**函数可以接受带空格的字符串

```
char str[50];
```

```
gets(str);
```

输入时: **I Love China!** ↵



6.3.6 善于使用字符串处理函数

3. **strcat**函数-----字符串连接函数

➤ 其一般形式为:

strcat(字符数组**1**, 字符数组**2**)

➤ 其作用是把两个字符串连接起来, 把字符串**2**接到字符串**1**的后面, 结果放在字符数组**1**中

使用字符串函数时,在程序开头用**#include <string.h>**



6.3.6 善于使用字符串处理函数

3. strcat函数-----字符串连接函数

char str1[30]="People"; 要足够大

char str2[]="China";

printf("%s", strcat(str1,str2));

输出: **PeopleChina**

建议: **strcat(str1,str2);**

printf("%s", str1);



6.3.6 善于使用字符串处理函数

4. strcpy和strncpy函数-字符串复制

➤ **strcpy**一般形式为:

strcpy(字符数组1,字符串2)

➤ 作用是将字符串**2**复制到字符数组**1**中去

```
char str1[10],str2[]="China";
```

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

str1	C	h	i	n	a	\0	\0	\0	\0	\0
------	---	---	---	---	---	----	----	----	----	----



6.3.6 善于使用字符串处理函数

4. strcpy和strncpy函数-字符串复制

➤ **strcpy**一般形式为:

strcpy(字符数组**1**,字符串**2**)

➤ 作用是将字符串**2**复制到字符数组**1**中去

char str1[10],str2[]="China";

strcpy(str1,str2); 要足够大

str1	C	h	i	n	a	\0	\0	\0	\0	\0
------	---	---	---	---	---	----	----	----	----	----



6.3.6 善于使用字符串处理函数

4. strcpy和strncpy函数-字符串复制

➤ **strcpy**一般形式为:

strcpy(字符数组1,字符串2)

➤ 作用是将字符串**2**复制到字符数组**1**中去

```
char str1[10],str2[]="China";
```

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

数组名形式



6.3.6 善于使用字符串处理函数

4. strcpy和strncpy函数-字符串复制

➤ **strcpy**一般形式为:

strcpy(字符数组1,字符串2)

➤ 作用是将字符串2复制到字符数组1中去

```
char str1[10],str2[]="China";
```

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

数组名或字符串常量



6.3.6 善于使用字符串处理函数

4. strcpy和strncpy函数-字符串复制

➤ **strcpy**一般形式为:

strcpy(字符数组1,字符串2)

➤ 作用是将字符串**2**复制到字符数组**1**中去

char str1[10],str2[]="China";

strcpy(str1,str2); 相当于

strcpy(str1,"China");



6.3.6 善于使用字符串处理函数

4. strcpy和strncpy函数-字符串复制

```
char str1[10],str2[]="China";
```

```
str1="China"; 错误
```

```
str1=str2; 错误
```



特别注意：字符串赋值、复制的方法 最容易出错的地方

char str1[10],str2[]="China"; 对

str1="China"; 错

str1= str2; 错

str1[0]=str2[0]; 对

str1[1]='a'; 对

str1[1]=str2; 错

strcpy(str1,str2); 对

strcpy(str1,"China"); 对

定义初始化
时，可用 =
其余不行



6.3.6 善于使用字符串处理函数

4. strcpy和strncpy函数-字符串复制

- 可以用**strncpy**函数将字符串**2**中前面**n**个字符复制到字符数组**1**中去
- **strncpy(str1, str2, 2);**
 - ◆ 作用是将**str2**中最前面**2**个字符复制到**str1**中，取代**str1**中原有的最前面**2**个字符
 - ◆ 复制的字符个数**n**不应多于**str1**中原有的字符



6.3.6 善于使用字符串处理函数

5. strcmp函数-----字符串比较函数

➤ 其一般形式为

strcmp(字符串1, 字符串2)

➤ 作用是比较字符串1和字符串2

➤ **strcmp(str1,str2);**

➤ **strcmp("China","Korea");**

➤ **strcmp(str1,"Beijing");**



6.3.6 善于使用字符串处理函数

5. strcmp函数----字符串比较函数

- 字符串比较的规则是：将两个字符串自左至右逐个字符相比，直到出现不同的字符或遇到'\0'为止
- 如全部字符相同，认为两个字符串相等
- 若出现不相同的字符，则以第一对不相同的字符的比较结果为准



6.3.6 善于使用字符串处理函数

5. strcmp函数-----字符串比较函数

"A"<"B"

"a">"A"

"computer">"compare"

"these">"that" "1A">"\$20"

"CHINA">"CANADA"

"DOG"<"cat"

"Tsinghua">"TSINGHUA"



6.3.6 善于使用字符串处理函数

5. strcmp函数----字符串比较函数

➤ 比较的结果由函数值带回

- ◆ 如果字符串 **1** = 字符串 **2**，则函数值为 **0**
- ◆ 如果字符串 **1** > 字符串 **2**，则函数值为一个正整数
- ◆ 如果字符串 **1** < 字符串 **2**，则函数值为一个负整数



6.3.6 善于使用字符串处理函数

5. strcmp函数----字符串比较函数

if(str1>str2) printf("yes"); 错误

if(strcmp(str1,str2)>0)

printf("yes"); 正确



6.3.6 善于使用字符串处理函数

if(str1==str2) printf("yes"); 错

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

printf("yes"); 正确

if(str1!=str2) printf("no"); 错

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

printf("no"); 正确



6.3.6 善于使用字符串处理函数

6. strlen函数-----测字符串长度的函数

➤ 其一般形式为：

strlen (字符数组)

➤ 它是测试字符串长度的函数

➤ 函数的值为字符串中的实际长度



6.3.6 善于使用字符串处理函数

6. **strlen**函数----测字符串长度的函数

```
char str[10]="China";  
printf("%d",strlen(str));
```

- 输出结果是**5**
- 也可以直接测试字符串常量的长度

```
strlen("China");
```



6.3.6 善于使用字符串处理函数

7. **strlwr**函数-----转换为小写的函数

➤ 其一般形式为

strlwr (字符串)

➤ 函数的作用是将字符串中大写字母换成小写字母



6.3.6 善于使用字符串处理函数

8. **strupr**函数-----转换为大写的函数

➤ 其一般形式为

strupr (字符串)

➤ 函数的作用是将字符串中小写字母换成大写字母



puts、**gets**、**strcat**、**strcpy**、
strcmp、**strlen** 等函数后面跟的参数，
为一维数组名，而不是数组元素，如

puts(str1); **puts(str1[0]);**(错)

strcmp(str1,str2);

strcmp(str1[0],str2[0]);(错)

printf 或 **scanf** 中控制字符为 **"%s"**

后面也是一维数组名，不加**&**，也不加**[]**

scanf("%s",str3)



C语言中的比较

注意：改错题常出这一类的题目

➤ 整数的比较，注意用`==` 不能用`=`

➤ 小数的比较，不能用`==`

用`fabs(a-b)<1e-6`

➤ 字符串的比较不能用`==` `<` `>` `!=`
要用函数`strcmp`

6.3.7 字符数组应用举例

例6.8 输入一行字符，统计其中有多少个单词，单词之间用空格分隔开。

- 解题思路：问题的关键是怎样确定“出现一个新单词了”
- ◆ 从第**1**个字符开始逐个字符进行检查，判断此字符是否是新单词的开头，如果是，就使变量**num**的值加**1**，最后得到的**num**的值就是单词总数



6.3.7 字符数组应用举例

能否判断空格数，单词数目为空格数加**1**

情况分析：字符分为两类，一类为空格，另一类为非空格。

空格 —> 非空格

空格 —> 空格

非空格 —> 非空格

非空格 —> 空格



6.3.7 字符数组应用举例

- ◆判断是否出现新单词，可以由是否有空格出现来决定(连续的若干个空格作为出现一次空格；一行开头的空格不统计在内)
- ◆如果测出某一个字符为非空格，而它的前面的字符是空格，则表示“新的单词开始了”，此时使**num**累加**1**
- ◆如果当前字符为非空格而其前面的字符也是非空格，则**num**不应再累加**1**



6.3.7 字符数组应用举例

即前一个字符为空格，当前字符为非空格，表示新单词开始。

解题思路：从字符串首个字符开始，循环依次逐个字符处理。

```
for (i=0;(c=string[i])!='\0';i++)
```

c表示当前处理的字符

此外再设置变量 **old**，表示前一个字符的值
old为空格，**c**为非空格时，表示新单词开始

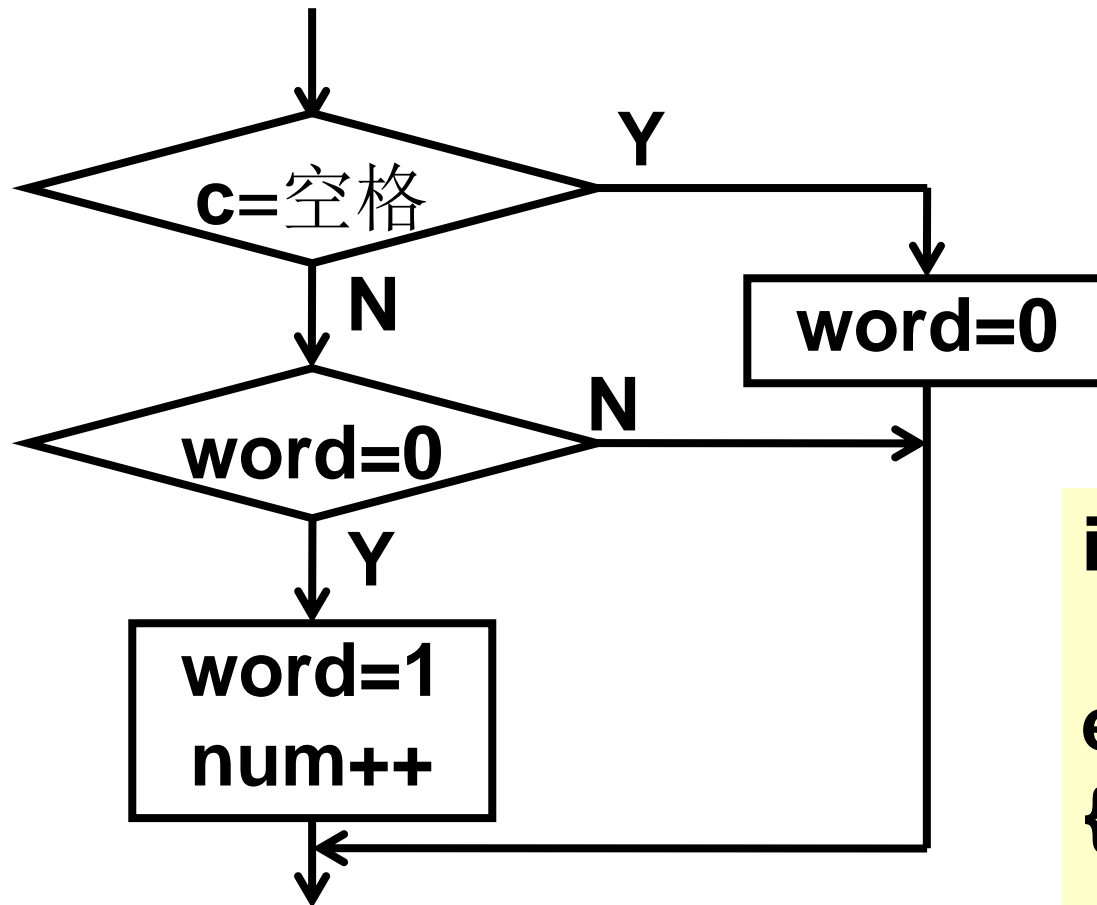



```
for (i=0;(c=string[i])!='\0';i++)  
{  
    if((old == ' ')&&(c!=' '))  
        {  
            num++;  
        }  
        old=c;  
}
```

6.3.7 字符数组应用举例

- ◆ 用变量**word**作为判别当前是否开始了一个新单词的标志，若**word=0**表示未出现新单词，如出现了新单词，就把**word**置成**1**
- ◆ 前面一个字符是否空格可以从**word**的值看出来，若**word**等于**0**，则表示前一个字符是空格；如果**word**等于**1**，意味着前一个字符为非空格





```
if(c==' ')  
    word=0;  
else if(word==0)  
{  
    word=1;  
    num++;  
}
```



当前字符	I		a	m		a		b	o	y	.
是否空格	否	是	否	否	是	否	是	否	否	否	否
word原值	0	1	0	1	1	0	1	0	1	1	1
新单词开始否	是	否	是	否	否	是	否	是	否	否	否
word新值	1	0	1	1	0	1	0	1	1	1	1
num值	1	1	2	2	2	3	3	4	4	4	4



.....

```
char string[81],c; int i,num=0,word=0;
```

```
gets(string);
```

```
for (i=0;(c=string[i])!='\0';
```

一定要设初始值

```
    if(c==' ') word=0;
```

```
    else if(word==0)
```

```
    { word=1;
```

```
        num++;
```

```
    }
```

```
printf("%d words\n",num);
```

.....



.....

```
char string[81],c; int i,num=0,word=0;
gets(string);
for (i=0;(c=string[i])!='\0';i++)
    if(c==' ') word=0;
    else if(word==0)
    { word=1;
      num++;
    }
printf("%d words\n",num);
```

.....

相当于
`c=string[i];`
`c!='\0'`

```
I am a boy.
4 words
```



对字符串处理的程序思路、框架

对字符串中的字符依次进行处理

```
for (i=0;(c=string[i])!='\0';i++)  
{  
    if(....).....  
    .....  
}
```

对字符串处理的程序思路、框架

对字符串中的字符依次进行处理

注意在末尾手工添加字符串结束标记'**\0**'

```
for (i=0;(c=string[i])!='\0';i++)
```

```
{
```

```
    if(....).....
```

```
    b[j]= c++.....
```

```
}
```

```
b[j+1]='\0';
```


例6.9 有**3**个字符串,要求找出其中最大者。

➤ 解题思路: 设一个二维的字符数组**str**,大小为**3×10**。每一行存放一个字符串

char str[3][10];



- 可以把**str[0],str[1],str[2]**看作**3**个一维字符数组，可以把它们如同一维数组那样进行处理

```
for (i=0;i<3;i++)  
    gets (str[i]);
```

```
China  
Japan  
India
```

str[0]	C	h	i	n	a	\0	\0	\0	\0	\0
str[1]	J	a	p	a	n	\0	\0	\0	\0	\0
str[2]	I	n	d	i	a	\0	\0	\0	\0	\0



➤ 经过三次两两比较,就可得到值最大者,把它放在一维字符数组**string**中

```
if (strcmp(str[0],str[1])>0)
```

```
    strcpy(string,str[0]);
```

```
else
```

```
    strcpy(string,str[1]);
```

```
if (strcmp(str[2],string)>0)
```

```
    strcpy(string,str[2]);
```



```
#include<stdio.h>
```

```
#include<string.h>
```

```
int main ( )
```

```
{char str[3][10]; char string[10]; int i;
```

```
for (i=0;i<3;i++) gets (str[i]);
```

```
if (strcmp(str[0],str[1])>0)
```

```
    strcpy(string,str[0]);
```

```
else
```

```
    strcpy(string,str[1]);
```

```
if (strcmp(str[2],string)>0)
```

```
    strcpy(string,str[2]);
```

```
printf("\nthe largest:\n%s\n",string);
```

```
return 0;
```

```
}
```

China

Japan

India

the largest:

Japan

