



中國人民大學

RENMIN UNIVERSITY OF CHINA

信息学院

SCHOOL OF INFORMATION

程序设计1荣誉课程

## 4. 数据结构1——数组/结构体/共用体

授课教师：游伟 副教授

授课时间：周一08:00 – 09:30, 周四16:00 – 17:30 (明德新闻楼0201)

上机时间：周四18:00 – 21:00 (理工配楼二层5号机房)

课程主页：<https://rucsesec.github.io/programming>

# 引子：生日祝福

【背景】为了在每位同学生生日时给同学献上祝福并且撰写人物传记，班委们不得不每天翻看通讯录寻找当日寿星。请编写一个小程序，帮助班委们完成寻找当日寿星的操作。具体而言，将每个同学在通讯录上的信息以一定组织形式存储，并在程序运行时遍历每个同学的生日信息，查看是否有当天生日的同学。

## 【思考】

- 每位同学的不同类型信息（学号、姓名、性别、生日等）如何组织在一起
- 所有同学的信息如何组织在一起
- 生日怎么存储才易于判断

# 目录

1. 一维数组
2. 二维数组
3. 多维数组
4. 字符数组
5. 结构体
6. 共同体

# 为什么需要数组

- 要向计算机输入全班50个学生一门课程的成绩

用50个float型简单变量表示学生的成绩

- 烦琐，如果有1000名学生怎么办呢？
- 没有反映出这些数据间的内在联系，实际上这些数据是同一个班级、同一门课程的成绩，它们具有相同的属性。

解决方法



数组

数组名  
下标  
→ s[15]

- (1) 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。
- (2) 用数组名和下标即可唯一地确定数组中的元素。
- (3) 数组中的每一个元素都属于同一个数据类型。

# 4.1 一维数组

## ■ 定义一维数组

类型说明符 数组名[常量表达式]

int a[10];

整型数组，即数组中的元素均为整型

数组名为a

数组包含10个整型元素

(1) 数组名的命名规则和变量名相同，遵循标识符命名规则。

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

相当于定义了10个简单的整型变量

(2) 在定义数组时，需要指定数组中元素的个数，方括号中的常量表达式用来表示元素的个数，即数组长度。

(3) 常量表达式中可以包括常量和符号常量，不能包含变量。

### 注意

- 数组元素的**下标从0开始**，用“int a[10];”定义数组，则最大下标值为9，不存在数组元素a[10]

## 4.1 一维数组

### ■ 引用一维数组元素

#### 数组名[下标]

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。

数组元素与一个简单变量的地位和作用相似。

“下标”可以是整型常量或整型表达式。

#### 注意

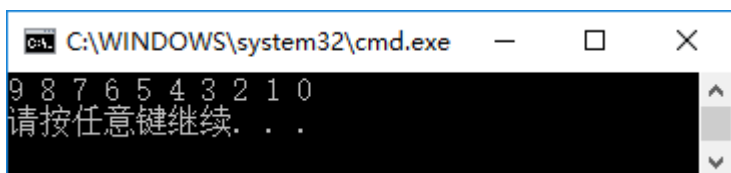
- 定义数组时用到的“数组名[常量表达式]”和引用数组元素时用的“数组名[下标]”在形式上相同，但含义不同。

```
int a[10];  
//前面有int,这是定义数组,指定数组包含10个  
//元素  
  
t=a[6];  
//这里的a[6]表示引用a数组中序号为6的元素
```

## 4.1 一维数组

■ 【例4-1】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

```
1. #include<stdio.h>
2. int main()
3. {
4.     int i,a[10];
5.     for(i=0; i<=9;i++)           //对数组元素a[0]~a[9]赋值
6.         a[i]=i;
7.     for(i=9;i>=0;i--)           //输出a[9]~a[0]共10个数组元素
8.         printf("%d ",a[i]);
9.     printf("\n");
10.    return 0;
11.}
```



```
C:\WINDOWS\system32\cmd.exe
9 8 7 6 5 4 3 2 1 0
请按任意键继续. . .
```



第1个for循环使a[0]~a[9]的值为0~9。

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

第2个for循环按a[9]~a[0]的顺序输出各元素的值。

## 4.1 一维数组

### ■ 一维数组的初始化

为了使程序简洁，常在定义数组的同时给各数组元素赋值，这称为数组的**初始化**。

(1) 在定义数组时对全部数组元素赋予初值。

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

将数组中各元素的初值顺序放在一对花括号内，数据间用逗号分隔。花括号内的数据就称为“**初始化列表**”。

(2) 可以只给数组中的一部分元素赋值。

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

定义a数组有10个元素，但花括号内只提供5个初值，这表示只给前面5个元素赋初值，系统自动给后5个元素赋初值为0。

(3) 给数组中全部元素赋初值为0。

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

或

```
int a[10]={0}; //未赋值的部分元素自动设定为0
```

(4) 在对全部数组元素赋初值时，由于数据的个数已经确定，因此可以不指定数组长度。

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

或

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

但是，如果数组长度与提供初值的个数不相同，则方括号中的数组长度不能省略。



回顾：21级图灵班选拔卷第四题【问题1】当输入为5时，下列代码的输出结果是多少？

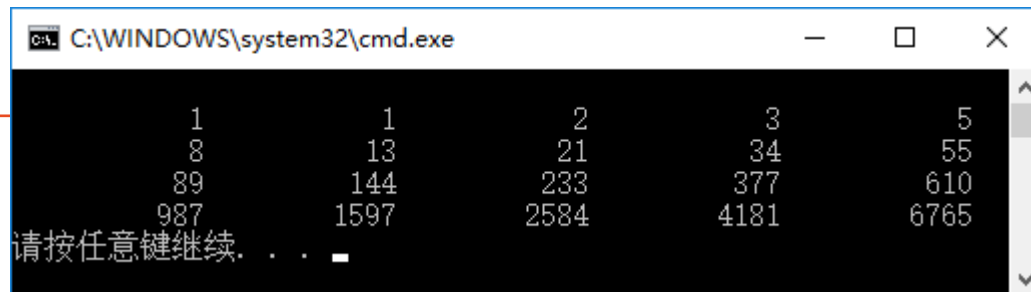
```
1. INPUT n
2. x←0
3. y←1
4. FOR i : 1 to n {
5.     z←ADD(x, y)
6.     x←y
7.     y←z
8. }
9. OUTPUT z
```

## 4.1 一维数组

答案：8（斐波拉契数列）

### ■ 【例4-2】用数组来处理求Fibonacci数列问题

```
1. #include <stdio.h>
2. int main()
3. {
4.     int i;
5.     int f[20]={0,1};           //对最前面两个元素f[0]和f[1]赋初值1
6.     for(i=2;i<20;i++)
7.         f[i]=f[i-2]+f[i-1];    //先后求出f[2]~f[19]的值
8.     for(i=0;i<20;i++)
9.     {
10.        if(i%5==0) printf("\n"); //控制每输出5个数后换行
11.        printf("%12d",f[i]);     //输出一个数
12.    }
13.    printf("\n");
14.    return 0;
15.}
```



```
C:\WINDOWS\system32\cmd.exe

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

请按任意键继续. ...
```

## 4.2 二维数组

### 小例子

有3个小分队，每队有6名队员，要把这些队员的工资用数组保存起来以备查。

	队员1	队员2	队员3	队员4	队员5	队员6
第1分队	2456	1847	1243	1600	2346	2757
第2分队	3045	2018	1725	2020	2458	1436
第3分队	1427	1175	1046	1976	1477	2018

如果建立一个数组pay，它应当是二维的，第一维用来表示第几分队，第二维用来表示第几个队员。例如用 $\text{pay}_{2,3}$ 表示2分队第3名队员的工资，它的值是1725。

二维数组常称为**矩阵**(matrix)。把二维数组写成**行**(row)和**列**(column)的排列形式，可以有助于形象化地理解二维数组的逻辑结构。

## 4.2 二维数组

### ■ 定义二维数组

类型说明符 数组名[常量表达式][常量表达式]

float pay[3][6];

float型二维数组

数组名为pay



float a[3][4], b[5][10];  
//定义a为3×4(3行4列)的数组, b为5×10(5行10列)的数组



float a[3, 4], b[5, 10]; //在一对方括号内不能写两个下标

数组第二维有6个元素

数组第一维有3个元素

二维数组可被看作一种特殊的一维数组: 它的元素又是一个一维数组。

例如, float a[3][4];可以把a看作一个一维数组, 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] —— a[0][0] a[0][1] a[0][2] a[0][3]

a[1] —— a[1][0] a[1][1] a[1][2] a[1][3]

a[2] —— a[2][0] a[2][1] a[2][2] a[2][3]

## 4.2 二维数组

### ■ 引用二维数组元素

数组名[下标][下标]

“下标”可以是整型常量或整型表达式。

数组元素可以出现在表达式中，也可以被赋值，如：  $b[1][2]=a[2][3]/2$ ;

#### 注意

- 在引用数组元素时，下标值应在已定义的数组大小的范围内。

```
int a[3][4]; //定义a为3×4的二维数组
```

```
a[3][4]=3; //不存在a[3][4]元素
```

//数组a可用的“行下标”的范围为0~2，  
“列下标”的范围为0~3



- 严格区分在**定义**数组时用的a[3][4]和**引用**元素时的a[3][4]的区别。
- 前者用a[3][4]来定义数组维数和各维大小
- 后者a[3][4]中的3和4是数组元素的下标值，a[3][4]代表行序号为3、列序号为4的元素（行序号和列序号均从0起算）。

## 4.2 二维数组

### ■ 二维数组的初始化

可以用“初始化列表”对二维数组初始化。

(1) 分行给二维数组赋初值。（最清楚直观）

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

(2) 可以将所有数据写在一个花括号内，按数组元素在内存中的排列顺序对各元素赋初值。

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

(3) 可以对部分元素赋初值。

```
int a[3][4]={{1},{5},{9}}; ①
```

```
int a[3][4]={{1},{0,6},{0,0,11}}; ②
```

```
int a[3][4]={{1},{5,6}}; ③
```

```
int a[3][4]={{1},{},{9}}; ④
```

①	②	③	④
1 0 0 0	1 0 0 0	1 0 0 0	1 0 0 0
5 0 0 0	0 6 0 0	5 6 0 0	0 0 0 0
9 0 0 0	0 0 11 0	0 0 0 0	9 0 0 0

(4) 如果对全部元素都赋初值(即提供全部初始数据)，则定义数组时对第1维的长度可以不指定，但第2维的长度不能省。

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

≡

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

在定义时也可以只对部分元素赋初值而省略第1维的长度，但应分行赋初值。

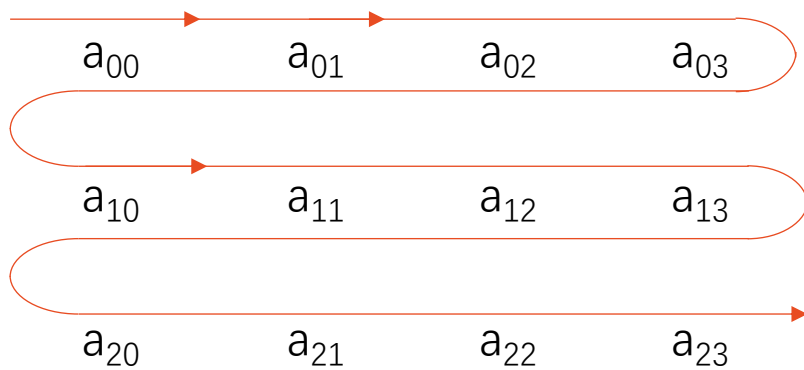
```
int a[][4]={{0,0,3},{},{0,10}};
```

## 4.2 二维数组

### ■ 二维数组的存储

C语言中，二维数组中元素排列的顺序是按行存放的。

```
float a[3][4]
```



#### 注意

- 用矩阵形式（如3行4列形式）表示二维数组，是逻辑上的概念，能形象地表示出行列关系。而在内存中，各元素是连续存放的，不是二维的，是线性的。

2000	$a[0][0]$	}
2004	$a[0][1]$	
2008	$a[0][2]$	
2012	$a[0][3]$	
2016	$a[1][0]$	}
2020	$a[1][1]$	
2024	$a[1][2]$	
2028	$a[1][3]$	
2032	$a[2][0]$	}
2036	$a[2][1]$	
2040	$a[2][2]$	
2044	$a[2][3]$	

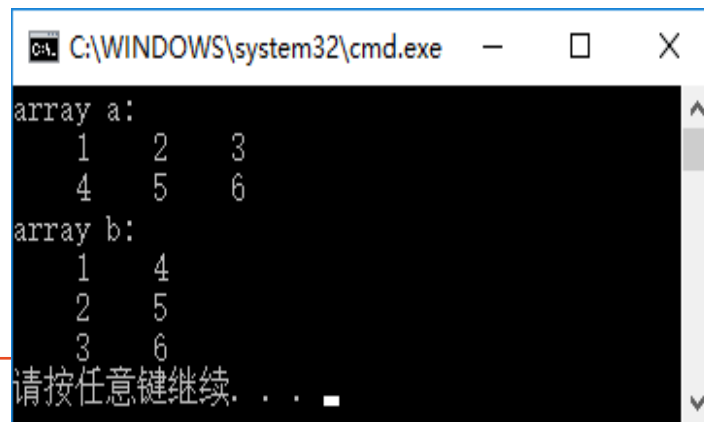
## 4.2 二维数组

■ 【例4-3】 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（矩阵转置）

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

```
1. #include <stdio.h>
2. int main()
3. {
4.     int a[2][3]={{1,2,3},{4,5,6}};
5.     int b[3][2],i,j;
6.     printf("array a:\n");
7.     for(i=0;i<=1;i++) //处理a数组一行中各元素
8.     {
9.         for (j=0;j<=2;j++) //处理a数组某一列中各元素
10.        {
11.            printf("%5d",a[i][j]); //输出一个元素
12.            //将a数组元素的值赋给b数组相应元素
13.            b[j][i]=a[i][j];
14.        }
15.        printf("\n");
16.    }
```

```
17.     printf("array b:\n");
18.     for(i=0;i<=2;i++) //处理b数组一行中各元素
19.     {
20.         for(j=0;j<=1;j++) //处理b数组某一列中各元素
21.             printf("%5d",b[i][j]); //输出b数组一个元素
22.         printf("\n");
23.     }
24.     return 0;
25. }
```



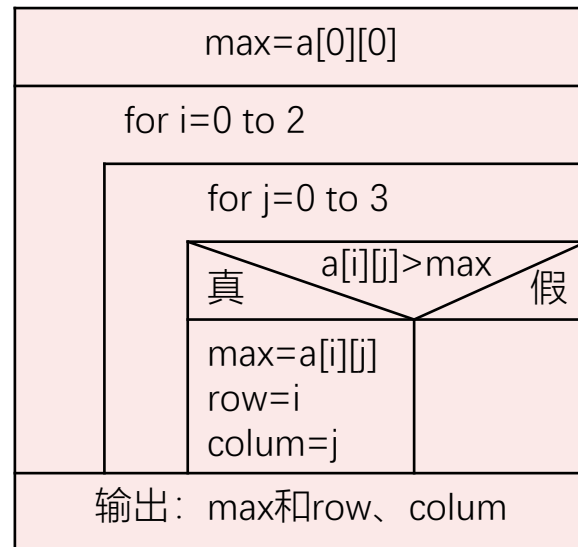
```
C:\WINDOWS\system32\cmd.exe
array a:
    1    2    3
    4    5    6
array b:
    1    4
    2    5
    3    6
请按任意键继续. . .
```

## 4.2 二维数组

■ 【例4-4】有一个 $3 \times 4$ 的矩阵，要求编程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。

```
1. #include <stdio.h>
2. int main()
3. {
4.     int i,j,row=0,column=0,max;
5.     int a[3][4]={{1,2,3,4},{9,8,7,6},{-10,10,-5,2}}; //定义数组并赋初值
6.     max=a[0][0]; //先认为a[0][0]最大
7.     for(i=0;i<=2;i++)
8.         for(j=0;j<=3;j++)
9.             if(a[i][j]>max) //如果某元素大于max, 就取代max的原值
10.            {
11.                max=a[i][j];
12.                row=i; //记下此元素的行号
13.                column=j; //记下此元素的列号
14.            }
15.     printf("max=%d\nrow=%d\ncolumn=%d\n",max,row,column);
16.     return 0;
17. }
```

先思考一下在打擂台时怎样确定最后的优胜者。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。再上去第3人，与台上的人(即刚才的得胜者)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过为止，最后留在台上的就是冠军。





## 4.3 多维数组

```
float a[2][3][4];    //定义三维数组a，它有2页，3行，4列
```

多维数组元素在内存中的排列顺序为: 第1维的下标变化最慢，第n维的下标变化最快。

---

float a[2, 3, 4];在内存中的排列顺序为:

a[0][0][0] → a[0][0][1] → a[0][0][2] → a[0][0][3] →

a[0][1][0] → a[0][1][1] → a[0][1][2] → a[0][1][3] →

a[0][2][0] → a[0][2][1] → a[0][2][2] → a[0][2][3] →

a[1][0][0] → a[1][0][1] → a[1][0][2] → a[1][0][3] →

a[1][1][0] → a[1][1][1] → a[1][1][2] → a[1][1][3] →

a[1][2][0] → a[1][2][1] → a[1][2][2] → a[1][2][3]

---

## 4.4 字符数组

用来存放字符数据的数组是**字符数组**。在字符数组中的一个元素内存放一个字符。

```
char c[10];  
c[0]='l'; 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]
l		a	m		h	a	p	p	y

由于字符型数据是以整数形式(ASCII代码)存放的，故也可以用整型数组来存放字符数据。

```
int c[10];  
c[0]='a';     //合法，但浪费存储空间
```

## 4.4.1 字符数组的初始化

对字符数组初始化，最容易理解的方式是用“初始化列表”，把各个字符依次赋给数组中各元素。

```
char c[10]={'l',' ','a','m',' ','h','a','p','p','y'}; //把10个字符依次赋给c[0] ~ c[9]这10个元素
```

如果在定义字符数组时不进行初始化，则数组中各元素的值是**不可预料**的。

如果花括号中提供的初值个数（即字符个数）大于数组长度，则出现语法错误。

如果初值个数小于数组长度，则只将这些字符赋给数组中前面那些元素，其余的元素自动定为空字符(即'\0')。

```
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

如果提供的初值个数与预定的数组长度相同，定义时可以省略数组长度，系统会自动根据初值个数确定长度。

```
char c[]={ 'l',' ','a','m',' ','h','a','p','p','y'}; //数组c的长度自动定为10
```

也可以定义和初始化一个二维字符数组。

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

```
      *
    * *
 *   *
 * *
 *
```

## 4.4.2 引用字符数组中的元素

### 【例4-5】输出一个已知的字符串

```
1. #include <stdio.h>
2. int main()
3. {
4.     char c[15]={'I',' ','a','m',' ','a',' ','
5.                 's','t','u','d','e','n','t','.'};
6.     int i;
7.     for(i=0;i<15;i++)
8.         printf("%c",c[i]);
9.     printf("\n");
10.    return 0;
11.}
```

### 【例4-6】输出一个菱形图

```
1. #include <stdio.h>
2. int main()
3. {
4.     char diamond[][5]={{' ',' ','*'},
5.                          {' ','*',' ','*'},
6.                          {'*',' ',' ',' ','*'},
7.                          {' ','*',' ','*'},
8.                          {' ',' ','*'}};
9.     int i,j;
10.    for (i=0;i<5;i++)
11.    {
12.        for (j=0;j<5;j++)
13.            printf("%c",diamond[i][j]);
14.        printf("\n");
15.    }
16.    return 0;
17.}
```

## 4.4.3 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。

在实际工作中，人们关心的往往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语言规定了一个“字符串结束标志”，以字符‘\0’作为结束标志。

“C program” 字符串是存放在一维数组中的，占10个字节，字符占9个字节，最后一个字节‘\0’是由系统自动加上的

### 注意

- C系统在用字符数组存储字符串常量时会自动加一个‘\0’作为结束符。
- 在定义字符数组时应估计实际字符串长度，保证数组长度始终大于字符串实际长度。
- 如果在一个字符数组中先后存放多个不同长度的字符串，则应使数组长度大于最长的字符串的长度。

## 4.4.3 字符串和字符串结束标志

```
printf("How do you do?\n");
```

在向内存中存储时，系统自动在最后一个字符'\n'的后面加了一个'\0'，作为字符串结束标志。在执行printf函数时，每输出一个字符检查一次，看下一个字符是否为'\0'，遇'\0'就停止输出。

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

或 

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

用一个字符串(注意字符串的两端是用**双引号**而不是单引号括起来的)作为字符数组的初值。

**注意**

- 数组c的长度不是10，而是11。因为字符串常量的最后由系统加上一个'\0'。

```
char c[]={'I', ' ', 'a', 'm', ' ', 'h', 'a', 'p', 'p', 'y', '\0'};    ≠
```

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

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

数组c的前5个元素为: 'C','h','i','n','a',第6个元素为'\0'，后4个元素也自动设定为空字符。

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

## 4.4.4 字符数组的输入输出

- (1) 逐个字符输入输出。用格式符 “%c” 输入或输出一个字符。
- (2) 将整个字符串一次输入或输出。用 “%s” 格式符，意思是对字符串(string)的输入输出。

```
1. #include <stdio.h>
2. int main()
3. {
4.     char c[15]={'I',' ','a','m',' ','a',' ','s','t','u','d','e','n','t','.'};
5.
6.     int i;
7.     for(i=0;i<15;i++)
8.         printf("%c",c[i]);
9.     printf("\n");
10.    return 0;
11.}
```

```
#include <stdio.h>
int main()
{
    char c[]="China";
    printf("%s\n",c);
    return 0;
}
```

- (1) 输出的字符中不包括结束符‘\0’。
- (2) 用“%s”格式符输出字符串时，printf函数中的输出项是字符数组名，而不是数组元素名。
- (3) 如果数组长度大于字符串的实际长度，也只输出到遇‘\0’结束。
- (4) 如果一个字符数组中包含一个以上‘\0’，则遇第一个‘\0’时输出就结束。



```
char c[6];  
scanf("%s",c);
```

China ✓

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

从键盘输入:

由于有空格字符分隔，作为3个字符串输入。

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

```
char str[13];
scanf("%s",str);
```

How are you? ↙

[illegible]



## 4.4.4 字符数组的输入输出

### 注意

- scanf函数中的输入项如果是字符数组名，**不要再加地址符&**，因为在C语言中数组名代表该数组第一个元素的地址(或者说数组的起始地址)。

```
scanf("%s", &str);
```

//str前面不应加&

- 若数组占6个字节。数组名c代表地址2000。可以用下面的输出语句得到数组第一个元素的地址。

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

//用八进制形式输出数组c的起始地址

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

//以字符串形式输出字符数组c的内容

c数组

2000	C
2001	h
2002	i
2003	n
2004	a
2005	\0

- 实际上是这样执行的：按字符数组名c找到其数组第一个元素的地址，然后逐个输出其中的字符，直到遇'\0'为止。

## 4.4.5 字符串处理函数

- 输出字符串的函数
- 输入字符串的函数
- 字符串连接函数
- 字符串复制函数
- 字符串比较函数
- 测字符串长度的函数
- 转换为大小写的函数

### 注意

- 字符串处理函数属于**库函数**。库函数并非C语言本身的组成部分，而是C语言编译系统为方便用户使用而提供的公共函数。不同的编译系统提供的函数数量和函数名、函数功能都不尽相同，使用时要小心，必要时查一下库函数手册。
- 在使用字符串处理函数时，应当在程序文件的开头用`#include <string.h>`把string.h文件包含到本文件中。

## 4.4.5 字符串处理函数

### ■ 输出字符串的函数

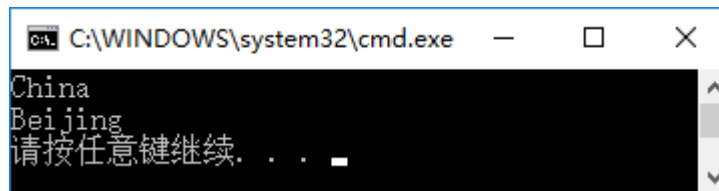
#### puts(字符数组)

作用：将一个字符串(以'\0'结束的字符序列)输出到终端。

用puts函数输出的字符串中可以包含转义字符。

在用puts输出时将字符串结束标志'\0'转换成'\n'，即输出完字符串后换行。

```
1. #include <stdio.h>
2. int main()
3. {
4.     char str[]={"China\nBeijing"};
5.     puts(str);
6.     return 0;
7. }
```



## 4.4.5 字符串处理函数

### ■ 输入字符串的函数

`gets(字符数组)`

作用：从终端输入一个字符串到字符数组，并且得到一个函数值。该函数值是字符数组的起始地址。

`gets(str);`

`//str是已定义的字符数组`

如果从键盘输入:

Computer↵

将输入的字符串“Computer”送给字符数组str（请注意，送给数组的共有9个字符，而不是8个字符），返回的函数值是字符数组str的第一个元素的地址。

**注意**

- 用puts和gets函数只能输出/输入一个字符串。



`puts(str1, str2); 或 gets(str1, str2);`

思考：以下代码的输出

## 4.4.5 字符串处理函数

### ■ 字符串连接函数

`strcat(字符串数组1, 字符串数组2)`

```
char str[100] = "China";  
strcat(str, str);  
printf("%s", str);
```

作用：把两个字符数组中的字符串连接起来，把字符串2接到字符串1的后面，结果放在字符数组1中，函数调用后得到一个函数值——字符数组1的地址。

字符数组1必须足够大，以便容纳连接后的新字符串。

连接前两个字符串的后面都有'\0'，连接时将字符串1后面的'\0'取消，只在新串最后保留'\0'。

```
char str1[30]={"People's Republic of "};  
char str2[]={"China"};  
printf("%s", strcat(str1, str2));
```

输出：People's Republic of China

连接前	str1:	P	e	o	p	l	e	'	s		R	e	p	u	b	l	i	c		o	f		\0	\0	\0	\0	\0	\0	\0	\0	\0
	str2:	C	h	i	n	a																									
连接后	str1:	P	e	o	p	l	e	'	s		R	e	p	u	b	l	i	c		o	f		C	h	i	n	a	\0	\0	\0	\0

## 4.4.5 字符串处理函数

### ■ 字符串复制函数

```
char str1[10], str2[]="China";  
strcpy(str1, str2); 或 strcpy(str1, "China");
```

**strcpy(字符数组1, 字符串2)**

执行后, str1: C h i n a \0 \0 \0 \0 \0

- 作用：将字符串2复制到字符数组1中去。
- 字符数组1必须定义得足够大，以便容纳被复制的字符串2。字符数组1的长度不应小于字符串2的长度。
- “字符数组1”必须写成数组名形式，“字符串2”可以是字符数组名，也可以是一个字符串常量。
- 若在复制前未对字符数组1初始化或赋值，则其各字节中的内容无法预知，复制时将字符串2和其后的'\0'一起复制到字符数组1中，取代字符数组1中前面的字符，未被取代的字符保持原有内容。
- 不能用赋值语句将一个字符串常量或字符数组直接给一个字符数组。字符数组名是一个地址常量，它不能改变值，正如数值型数组名不能被赋值一样。

```
str1="China"; str1=str2;
```
- 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);
```
- 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。但复制的字符个数n不应多于str1中原有的字符（不包括'\0'）。

## 4.4.5 字符串处理函数

### ■ 字符串比较函数

**strcmp(字符串1, 字符串2)**

- 作用：比较字符串1和字符串2。
- 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(按ASCII码值大小比较)，直到出现不同的字符或遇到'\0'为止。
  - (1) 如全部字符相同，则认为两个字符串相等；
  - (2) 若出现不相同的字符，则以第1对不相同的字符的比较结果为准。
- 比较的**结果**由函数值带回。
  - (1) 如果字符串1与字符串2相同，则函数值为0。
  - (2) 如果字符串1>字符串2，则函数值为一个正整数。
  - (3) 如果字符串1<字符串2，则函数值为一个负整数。

#### 注意

- 对两个字符串比较不能直接用`str1>str2`进行比较，因为`str1`和`str2`代表地址而不代表数组中全部元素，而只能用`(strcmp(str1, str2)>0)`实现，系统分别找到两个字符数组的第一个元素，然后顺序比较数组中各个元素的值。

## 4.4.5 字符串处理函数

### ■ 测字符串长度的函数

#### strlen(字符数组)

作用：测试字符串长度的函数。函数的值为字符串中的实际长度(不包括'\0'在内)。

```
1. #include <stdio.h>
2. #include <string.h>
3. int main()
4. {
5.     char str[10]="China";
6.     printf("%d,%d\n",strlen(str),sizeof(str));
7. }
```

### ■ 转换为大小写的函数

#### strlwr(字符串)

作用：将字符串中大写字母换成小写字母。

#### strupr(字符串)

作用：将字符串中小写字母换成大写字母。



## 4.4.6 字符数组应用举例

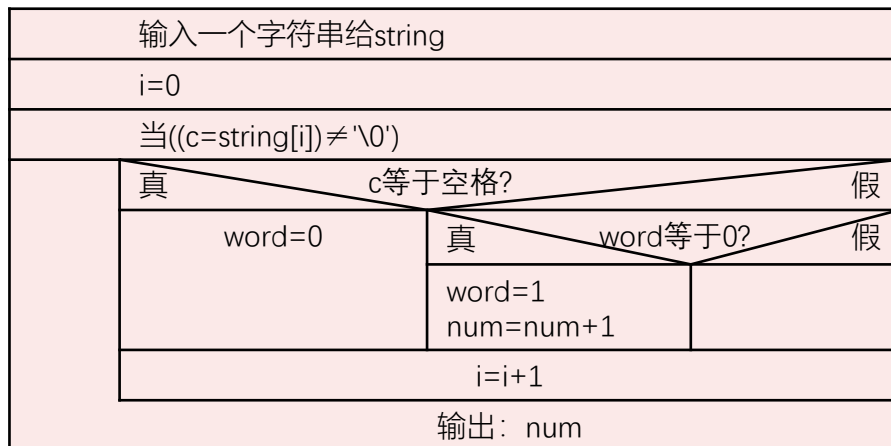
### ■ 【例4-7】输入一行字符，统计其中有多少个单词，单词之间用空格分隔开

string: 用于存放字符串。

i: 计数器，用于遍历字符串中的每个字符。

word: 用于判断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。

num: 用于统计单词数。



```
1. #include <stdio.h>
2. int main()
3. {
4.     char string[81];
5.     int i,num=0,word=0;
6.     char c;
7.     gets(string);    //输入一个字符串给字符数组
8.     for(i=0;(c=string[i])!='\0';i++) //字符'\0'结束循环
9.         if(c==' ') word=0; //若是空格字符，使word置0
10.        else if(word==0)    //若不是空格字符且word原值为0
```

```
11.     {
12.         word=1;    //使word置1
13.         num++;    //num累加1，表示增加一个单词
14.     }
15.     printf("There are %d words in this line.\n",num);
16.     return 0;
17. }
```

## 4.4.6 字符数组应用举例

### ■ 【例4-8】有3个字符串,要求找出其中“最大”者

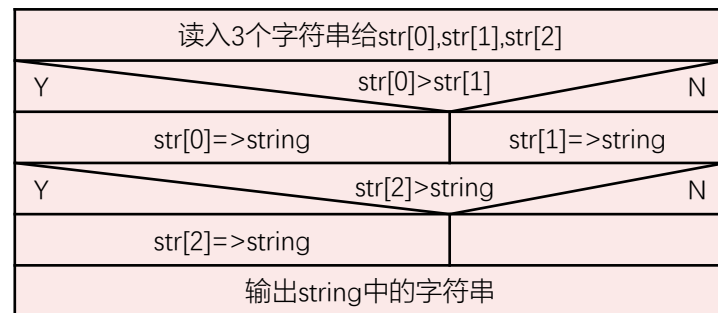
str[0]:	H	o	l	l	a	n	d	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0
str[1]:	C	h	i	n	a	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0
str[2]:	A	m	e	r	i	c	a	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0	\0

```
1. #include<stdio.h>
2. #include<string.h>
3. int main()
4. {
5.     char str[3][20];    //定义二维字符数组
6.     char string[20];    //定义一维字符数组,作为交换字符串时的临时字符数组
7.     int i;
8.     for(i=0;i<3;i++) gets(str[i]); //读入3个字符串

9.     if(strcmp(str[0],str[1])>0)    //若str[0]大于str[1]
10.        strcpy(string,str[0]);    //把str[0]的字符串赋给字符数组string
11.     else
12.        strcpy(string,str[1]);    //把str[1]的字符串赋给字符数组string

13.     if(strcmp(str[2],string)>0)    //若str[2]大于string
14.        strcpy(string,str[2]);    //把str[2]的字符串赋给字符数组string

15.     printf("\nthe largest string is:\n%s\n",string); //输出string
16.     return 0;
17. }
```



(1) 流程图和程序注释中的“大于”是指两个字符串的比较中的“大于”。

(2) str[0], str[1], str[2]和string是一维字符数组,其中可以存放一个字符串。

(3) strcpy函数在将str[0], str[1]或str[2]复制到string时,最后都有一个'\0'。因此,最后用%s格式输出string时,遇到string中第一个'\0'即结束输出,并不是把string中的全部字符输出。