



数组

程 淼

E-mail: mew_cheng@outlook.com

数组

- ✧ 对一系列的同类型数据进行处理
 - ✧ 一个班有30个学生，每个学生有一个成绩，要求这30名学生的平均成绩
 - ✧ 学生1的成绩+学生2的成绩+ ... + 学生30的成绩

数组



✧ 一批具有同名的同属性的数据就组成一个数组



数组

- ✧ 数组是一组有序数组的集合
- ✧ 用一个数组名来唯一地确定数组中的元素
- ✧ 数组中的每一个元素都属于同一个数据类型
 - ✧ 不能把不同类型的数据(如学生的成绩和学生的性别)放在同一个数组中

一维数组

✧ 对一维数组的定义：

✧ `int a[10];`

✧ 一般形式：

✧ 类型符 数组名[常量表达式];

一维数组

- ✧ 数组名的命名规则和变量名相同，遵循标识符命名规则
- ✧ 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
- ✧ 常量表达式中可以包括常量和符号常量
 - ✧ `int a[3+5];` 合法
 - ✧ `int a[n];` 不合法

一维数组

- ✧ 在内存中划出一片存储空间，存放了一个有10个整型元素的数组

a数组

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

一维数组

- ✧ 如果在被调用的函数（不包括主函数）中定义数组，其长度可以是变量或非常量表达式

```
void func(int n)
{
    int a[2*n];
}
```

- ✧ 如果指定数组为静态(static)存储方式，则不能用“可变长数组”

```
static int a[2*n];
```


一维数组

- ✧ 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
 - ✧ 数组名[下标]
 - ✧ `a[0]`就是数组`a`中序号为0的元素，它和一个简单变量的地位和作用相似
 - ✧ `int a[10];`
 - ✧ `t = a[6];`

一维数组

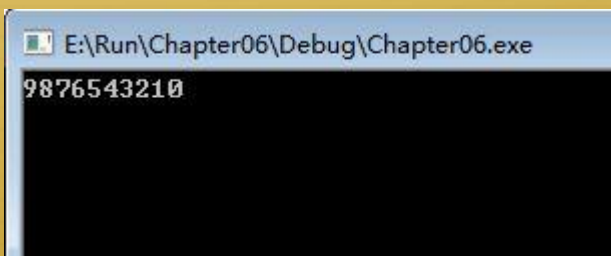
- ✧ 对10个数组元素依次赋值为0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 要求按逆序输出

```
#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;
}
```



一维数组

✧ 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};`

✧ 可以只给数组中的一部分元素赋值

✧ `int a[10] = {0, 1, 2, 3, 4};`

✧ 如果想使一个数组中全部元素值为0，可以写成

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

一维数组

- ✧ 在对全部数组元素赋初值时，如果数据的个数已经确定，因为可以不指定数组长度
 - ✧ `int a[5] = {1, 2, 3, 4, 5};`
 - ✧ `int a[] = {1, 2, 3, 4, 5};`
- ✧ 凡未被“初始化列表”指定初始化的数组元素，系统会自动把它们初始化为0

一维数组

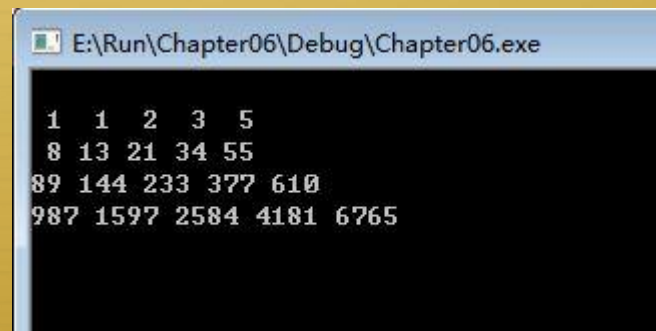
✧ 用数组来处理求Fibonacci数列问题

```
#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("%2d ", f[i]);
    }

    printf("\n");
    return 0;
}
```

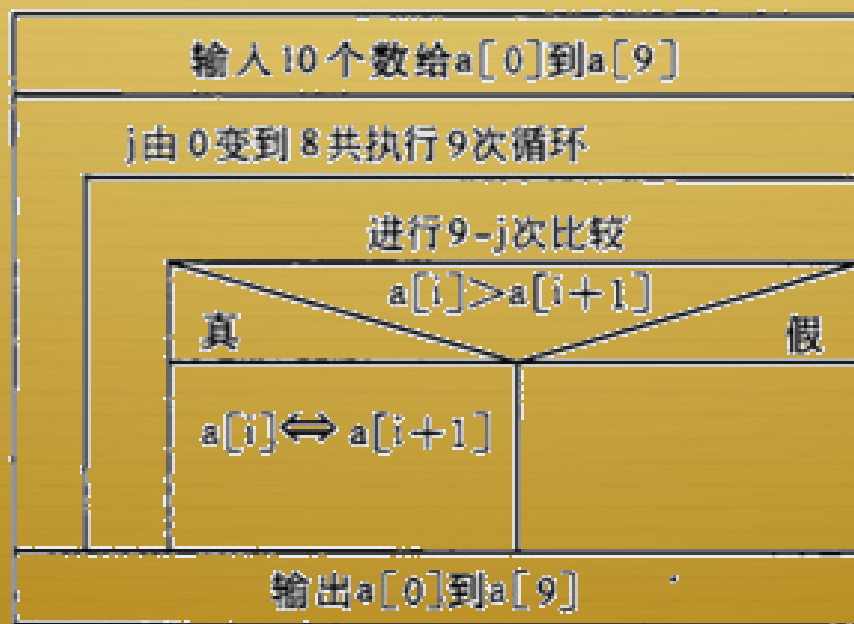


```
E:\Run\Chapter06\Debug\Chapter06.exe

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


一维数组

✧ 有10个地区的面积，要求对它们按由小到大的顺序排列



一维数组

✧ 有10个地区的面积，要求对它们按由小到大的顺序排列

```
#include <stdio.h>

int main()
{
    int a[10];
    int i, j, t;
    printf("input 10 numbers : \n");

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

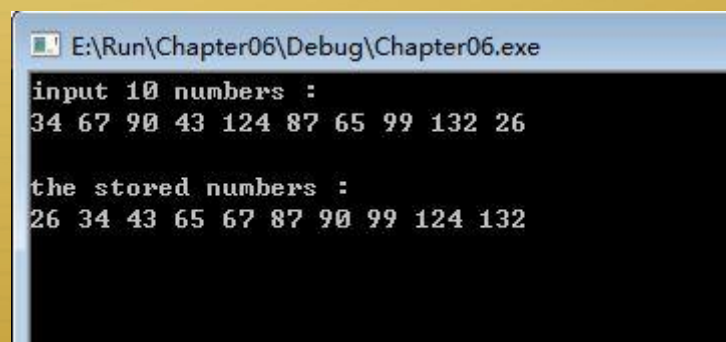
    printf("\n");

    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 stored numbers : \n");

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

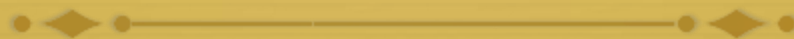
    printf("\n");
    return 0;
}
```



```
E:\Run\Chapter06\Debug\Chapter06.exe
input 10 numbers :
34 67 90 43 124 87 65 99 132 26

the stored numbers :
26 34 43 65 67 87 90 99 124 132
```

二维数组



二维数组

- ✧ 数组也可以是二维的，此时数组也称为矩阵(matrix)
- ✧ 二维数组写成行(column)和列(row)的排列形式



二维数组

✧ 二维数组的表示形式:

✧ 数组名[下标][下标]

✧ 二维数组的定义:

✧ `float pay[3][6];`

✧ 一般形式:

✧ 类型说明符 数组名 [常量表达式][常量表达式]

✧ `float a[3][4], b[5][10]`

✧ `float a[3, 4], b[5, 10]`

二维数组

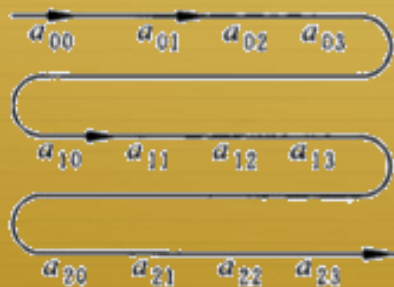
- ✧ C语言可将二维数组看作是一种特殊的一维数组;
- ✧ 它的元素又是一个一维数组

一维数	{	a[0]	----	a[0][0]	a[0][1]	a[0][2]	a[0][3]
组的名		a[1]	----	a[1][0]	a[1][1]	a[1][2]	a[1][3]
字		a[2]	----	a[2][0]	a[2][1]	a[2][2]	a[2][3]

二维数组

- ✧ C语言中，**二维数组中的元素排列的顺序是按行存放的**，即在内存中先顺序存放第1行的元素，接着再存放第2行的元素

✧ $a[3][4]$



2000	a[0][0]	第 0 行元素
2004	a[0][1]	
2008	a[0][2]	
2012	a[0][3]	
2016	a[1][0]	第 1 行元素
2020	a[1][1]	
2024	a[1][2]	
2026	a[1][3]	
2032	a[2][0]	第 2 行元素
2036	a[2][1]	
2040	a[2][2]	
2044	a[2][3]	

二维数组

✧ 多维数组元素内存中的排列也是顺序的

✧ `float a[2][3][4];`

`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]`

二维数组

- ✧ 数组元素可以出现在表达式中，也可以被赋值
 - ✧ $b[1][2]=a[2][3]/2$
- ✧ 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];`
 - ✧ `a[3][4] = 3;`

二维数组

✧ 可以用“初始化列表”分行对二维数组初始化

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

✧ 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};`

二维数组

✧ 可以对部分元素赋初值

✧ `int a[3][4] = {{1}, {5}, {9}};`

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

✧ 也可以对各行中的某一元素赋初值

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

1	0	0	0
0	6	0	0
0	0	11	0

二维数组

✧ 可以只对某几行元素赋初值

✧ `int a[3][4] = {{1}, {5, 6}};`

✧ 第3行不赋初值

✧ 也可以对第2行不赋初值

✧ `int a[3][4] = {{1}, {}, {9}};`

二维数组

✧ 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}};`

0	0	3	0
0	0	0	0
0	10	0	0

二维数组

- ✧ 一个例子：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。例如：

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

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

二维数组

✧ 编写程序

```
#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++)
    {
        for (j = 0; j <= 2; j++)
        {
            printf("%5d", a[i][j]);
            b[j][i] = a[i][j];
        }
        printf("\n");
    }
    printf("array b: \n");

    for (i = 0; i <= 2; i++)
    {
        for (j = 0; j <= 1; j++)
            printf("%5d", b[i][j]);

        printf("\n");
    }

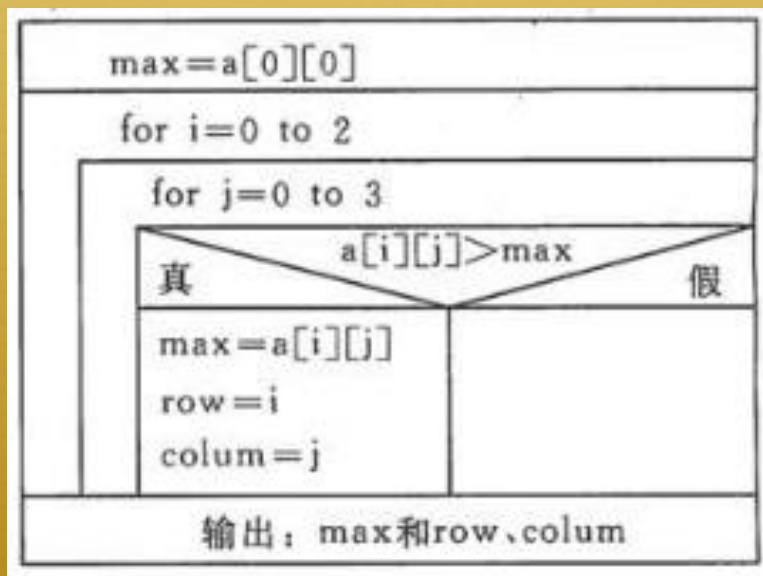
    return 0;
}
```

```
E:\Run\Chapter06\Debug\Chapter06.exe
array a:
    1    2    3
    4    5    6
array b:
    1    4
    2    5
    3    6
```


二维数组

✧ 一个例子：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



二维数组

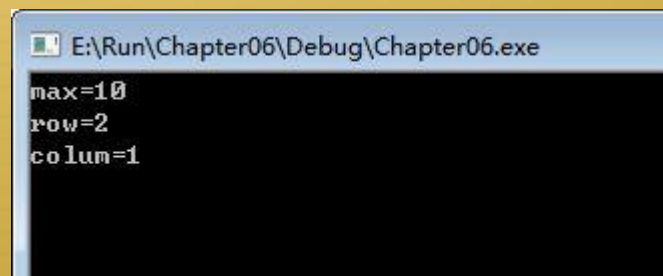
✧ 编写程序

```
#include <stdio.h>

int main()
{
    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];

    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\ncolumn=%d\n", max, row, column);
    return 0;
}
```



E:\Run\Chapter06\Debug\Chapter06.exe

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

字符数组

- ✧ 字符型数据是以字符的ASCII代码存储在存储单元中的，一般占一个字节。
- ✧ C语言中没有字符串类型，字符串是存放在字符型数组中的
- ✧ 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。

字符数组

✧ 字符数组的定义:

✧ `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

字符数组

✧ 用“初始化列表”对字符数组进行初始化

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

✧ 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，其余的元素自动定位空字符(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

字符数组

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

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

✧ 数组的长度自动定为10

✧ 定义初始化一个二维字符数组

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

✧ `{ '*', ' ', ' ', ' ', '*' }, { ' ', '*', ' ', '*' }, { ' ', ' ', '*' } };`

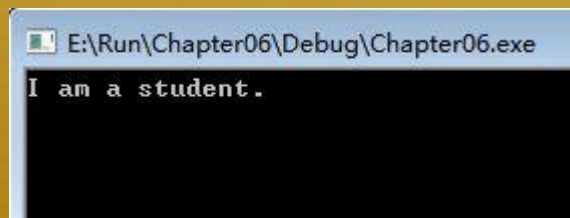
字符数组

- ✧ 可以引用字符数组中的一个元素，得到一个字符
- ✧ 一个例子：输出一个已知的字符串

```
#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;
}
```



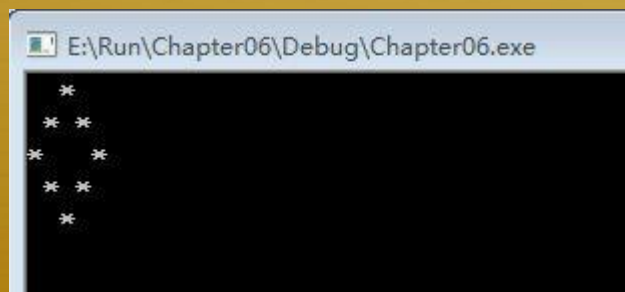
字符数组

✧ 一个例子：输出一个菱形图

```
#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;
}
```



字符串结束标志

- ✧ 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
- ✧ 在遇到字符 ‘\0’ 时，表示字符串结束，把它前面的字符组成一个字符串
- ✧ 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。
- ✧ 例如，“C program” 共有9个字符。字符串是存放在一堆数组中的，在数组中它占10个字节，最后一个字节 ‘\0’ 是由系统自动加上的

字符串结束标志

- ✧ ‘\0’代表ASCII码为0的字符，从ASCII码表中可以查到，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
- ✧ `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 ‘\0’，遇 ‘\0’就停止输出

字符串结束标志

✧ 用字符串常量来使字符数组初始化

✧ `char c[] = {"I am happy"};`

✧ `char c[] = "I am happy";`

✧ 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'};`

✧ 如果有 `char c[10] = {"China"}`

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

字符串结束标志

- ✧ 从键盘输入“Hello”分别赋给c数组中前面5个元素

C		p	r	o	g	r	a	m	.	\0
---	--	---	---	---	---	---	---	---	---	----

H	e	l	l	o	g	r	a	m	.	\0
---	---	---	---	---	---	---	---	---	---	----

- ✧ 如果想输出字符数组中的字符串，则会连续输出Helloqram.
- ✧ 如果在“Hello”后面加一个'\0'，它取代了第6个字符“g”。在数组中的存储情况为

H	e	l	l	o	\0	r	a	m	.	\0
---	---	---	---	---	----	---	---	---	---	----

- ✧ 只输出了“Hello”

字符数组的输入输出

- ✧ 逐个字符输入输出

- ✧ 用格式符 “%c”输入或输出一个字符

- ✧ 将整个字符串一次输入或输出

- ✧ 用 “%s”格式符，对字符串(string)的输入输出

- ✧ 例如：

- ✧ `char c[] = {"China"};`

- ✧ `printf("%s\n", c);`

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

字符数组的输入输出

- ✧ 输出的字符串不包括结束符'\0'
- ✧ 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
- ✧ 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
 - ✧ `char c[10] = {"China"};`
 - ✧ `printf("%s", c);`

字符数组的输入输出

- ✧ 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
- ✧ 可以用scanf函数输入一个字符串
 - ✧ scanf(“%s”, c);
 - ✧ 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
 - ✧ 例如，char c[6]，从键盘输入 China

字符数组的输入输出

✧ 多个字符串的输入

✧ `char str1[5], str2[5], str3[5];`

✧ `scanf("%s %s %s", str1, str2, str3);`

H	o	w	\0	\0
a	r	e	\0	\0
y	o	u	?	\0

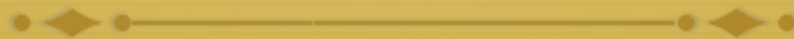
字符数组的输入输出

✧ 输出字符串的方法

✧ `printf("%s", c);`

✧ 按字符数组名`c`找到其数组起始地址，然后逐个输出其中的字符，直到遇 ‘`\0`’为止

字符串处理函数



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
- ✧ 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>`

字符串处理函数

- ✧ puts函数-----输出字符串的函数
- ✧ puts(字符数组): 将一个字符串输出到终端
 - ✧ puts(str);
 - ✧ 例如, char str[] = {"China \nBeijing"}; puts(str);
- ✧ 由于可以用printf函数输出字符串, 因此puts函数用得不多

字符串处理函数

✧ gets函数-----输入字符串的函数

✧ 一般形式为:

✧ gets(字符数组): 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);

✧ 从键盘输入: Computer

✧ char str[] = "Computer";

✧ 用puts和gets函数只能输出或输入一个字符串

✧ puts(str1, str2);

✧ gets(str1, str2);

字符串处理函数

✧ strcat函数-----字符串连接函数

✧ 一般形式为:

✧ strcat(字符数组1, 字符数组2)

✧ strcat是STRing CATenate的缩写

字符串处理函数

- ✧ `strcat`函数，其作用是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

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

输出：

People's Republic of China

- ✧ 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’

字符串处理函数

✧ strcpy和strncpy函数-----字符串复制函数

✧ strcpy(字符数组1,字符串2): 将字符串2复制到字符数组1中去

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

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

字符串处理函数

✧ strcpy和strncpy函数

- ✧ 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
 - ✧ strcpy(str1, “China”);
- ✧ 如果在复制前未对str1数组初始化或赋值，则str1各字节中的内容是无法预知的
 - ✧ 复制时将str2中的字符串和其后的 ‘\0’一起复制到字符数组1中，取代字符数组1中的前面6个字符
 - ✧ 最后4个字符并不一定是 ‘\0’，而是str1中原有的最后4个字节的内容

字符串处理函数

- ✧ 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

```
str1 = "China";    ✖  
Str1 = str2;       ✖
```

- ✧ 只能用strcpy函数将一个字符串复制到另一个字符数组中去
- ✧ 赋值语句只能将一个字符赋给一个字符型变量或字符数组元素

```
char a[5], c1, c2;  
c1 = 'A'; c2 = 'B';  
a[0] = 'C'; a[1] = 'h'; a[2] = 'i'; a[3] = 'n'; a[4] = 'a';
```

字符串处理函数

- ✧ 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
 - ✧ 复制的字符个数n不应多于str1中原有的字符(不包括'\0')

字符串处理函数

✧ strcmp函数-----字符串比较函数

✧ strcmp(字符串1, 字符串2)

✧ 比较字符串1和字符串2

```
strcmp(str1, str2);  
strcmp("China", "Korea");  
strcmp(str1, "Beijing");
```

✧ 小写字母比大写字母“大”，所以“DOG” < “cat”

字符串处理函数

✧ strlen函数-----测字符串长度

✧ 一般形式：strlen(字符数组)

```
char str[10] = "China";  
printf("%d", strlen(str));  
输出结果是5
```

✧ strlwr函数-----转换为小写的函数

✧ 一般形式：strlwr(字符串)

✧ 将字符串中大写字母换成小写字母

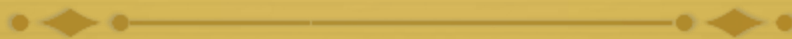
字符串处理函数

✧ `strupr`函数-----转换为大写的函数

✧ 一般形式: `strupr(字符串)`

✧ 将字符串中小写字母换成大写字母

字符数组应用举例



字符数组应用举例

- ✧ 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：判断是否出现新单词，可以由是否有空格出现来决定
- ✧ 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”
- ✧ 如果当前字符为非空格而其前面的字符也是非空格，则意味着仍然是原来那个单词的继续

字符数组应用举例

✧ 解题思路

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

字符数组应用举例

✧ 解题思路

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

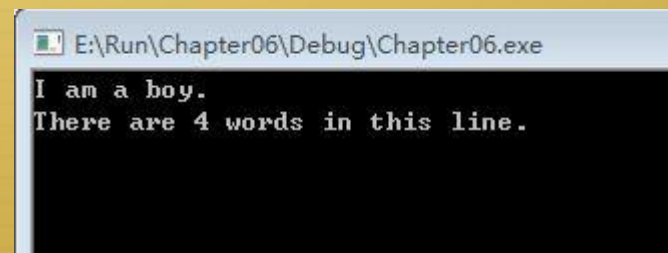
字符数组应用举例

✧ 编写程序

```
#include <stdio.h>

int main()
{
    char string[81];
    int i, num = 0, word = 0;
    char c;
    gets(string);
    for (i = 0; (c=string[i]) != '\0'; i++)
        if (c == ' ') word = 0;
        else if (word == 0)
        {
            word = 1;
            num++;
        }

    printf("There are %d words in this line. \n", num);
    return 0;
}
```



E:\Run\Chapter06\Debug\Chapter06.exe

I am a boy.
There are 4 words in this line.

字符数组应用举例

- ✧ 有3个字符串，要求找出其中最大者
- ✧ 解题思路：可以设一个二维的字符数组str，大小为3*20，即有3行20列
 - ✧ 用gets函数分别读入3个字符串，赋给3个一维字符数组
 - ✧ 经过3次两两比较，就可得到值最大者

字符数组应用举例

✧ 编写程序

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

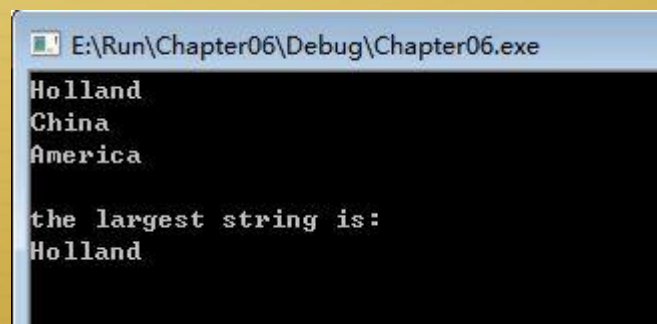
int main()
{
    char str[3][20];
    char string[20];
    int i;

    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("the largest string is: %s\n", string);
    return 0;
}
```



```
E:\Run\Chapter06\Debug\Chapter06.exe
Holland
China
America

the largest string is:
Holland
```

