

Welcome

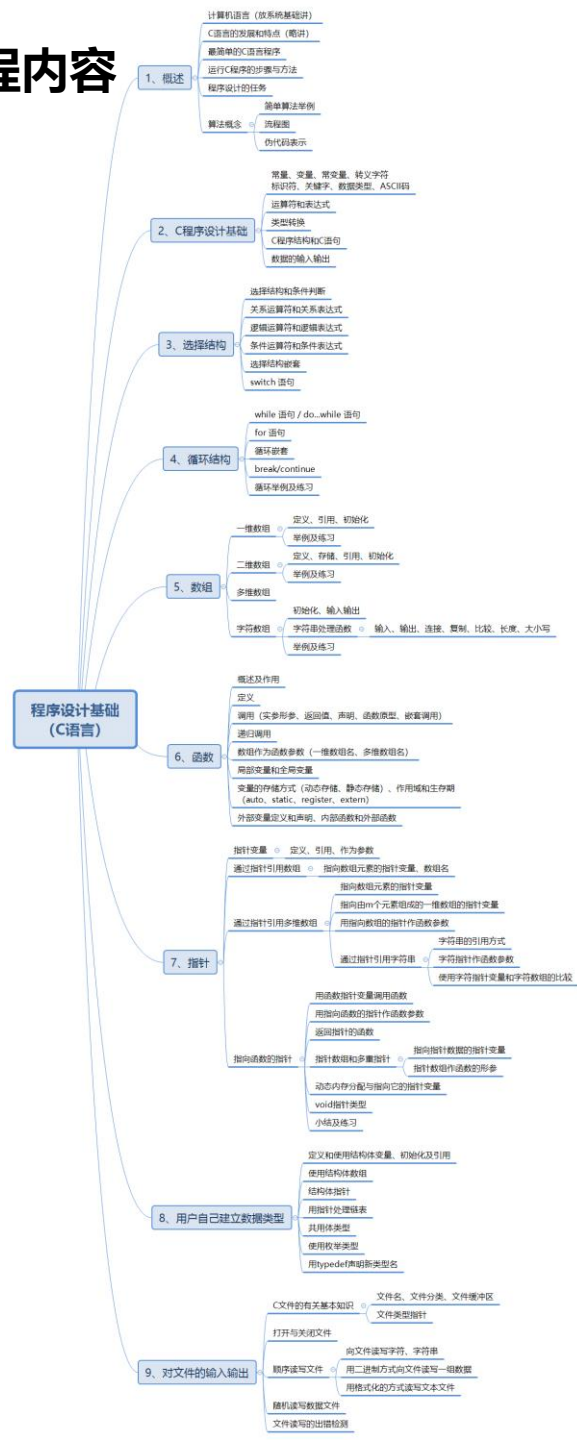
数据科学与大数据技术专业 程序设计基础(C语言)

上海体育学院经济管理学院

Wu Ying



课程内容



第5周 程序设计概述

计算机语言及C语言

最简单的C语言程序

运行C程序的步骤和方法

程序设计的任务

算法概念及简单算法

算法表示

流程图

伪代码

第6周 C程序设计基础

常量、变量及数据类型

常量变量常量变量转义字符

标识符关键字数据类型

运算符和表达式及类型转换

C程序结构和C语句

数据的输入输出

第7周 选择结构

选择结构和条件判断

运算符和表达式

关系运算符和关系表达式

逻辑运算符和逻辑表达式

条件运算符和条件表达式

选择结构嵌套

switch 语句

第8周 循环结构

while语句和do...while语句

for语句

循环嵌套, break, continue

循环练习

第9周 数组 (一)

一维数组定义、引用、初始化

一维数组练习

二维数组定义引用初始化

多维数组及练习

第10周 数组 (二) 及函数 (一)

字符数组初始化及输入输出

字符串处理函数

函数定义与调用

函数返回值、声明、原型、嵌套调用

第11周 函数 (二)

递归调用

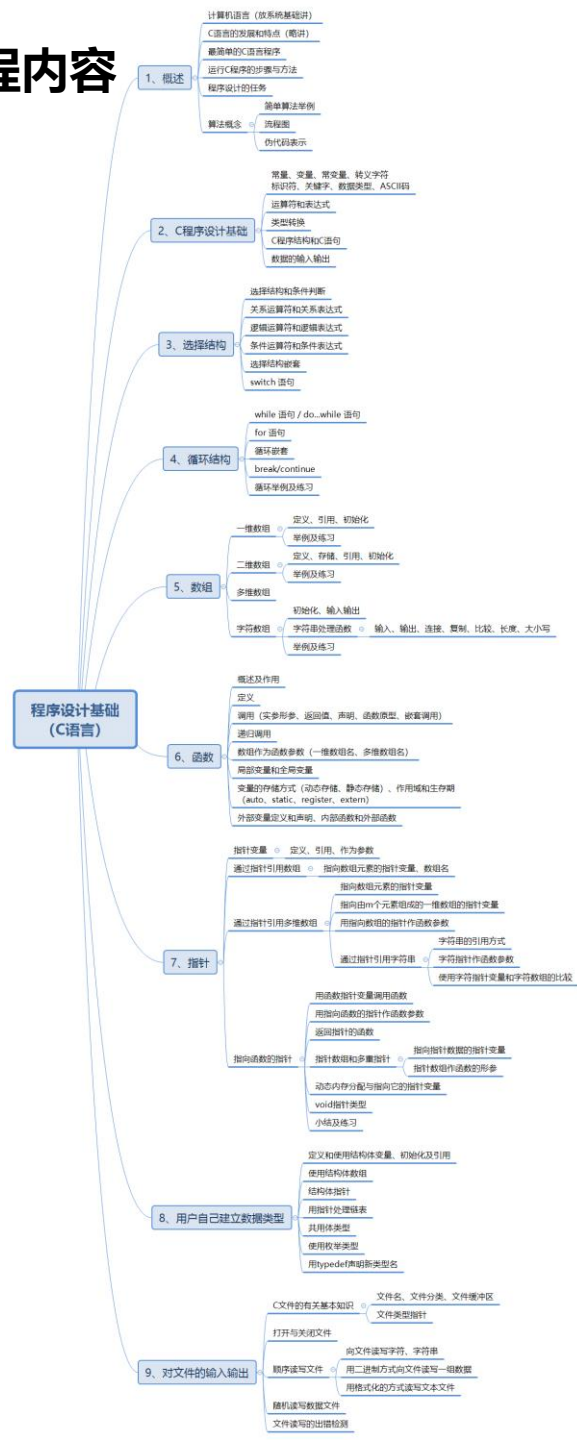
数组作为函数参数

变量作用域和生存期

变量存储方式、内部函数与外部函数



课程内容



第12周 指针 (一)

指针变量

通过指针引用数组

通过指针引用多维数组 (I)

指向数组元素的指针变量

指向一维数组的指针变量

第13周 指针 (二)

通过指针引用多维数组 (II)

用指向数组的指针作为函数参数

通过指针引用字符串

字符串的引用方式

字符指针作函数参数

使用字符指针变量和字符串数组的比较

第14周 指针 (三)

指向函数的指针 (I)

用函数指针变量调用函数

用指向函数的指针作函数参数

返回指向指针的函数

指向函数的指针 (II)

指针数组和多重指针

指向指针数据的指针变量

指针数组作函数的形参

动态内存分配与指向它的指针变量

第15周 指针 (四) 及结构体 (一)

指向函数的指针 (IV)

void指针类型

小结及练习

结构体

定义和使用结构体变量

使用结构体数组、结构体指针

第16周 结构体 (二) 及文件 (一)

用指针处理链表、共用体类型

枚举类型、使用typedef声明新类型

C文件的基本知识

打开与关闭文件

第17周 文件 (二)

顺序读写文件

随机读写数据文件

总复习练习

总复习练习

二维数组的初始化

(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}};
```

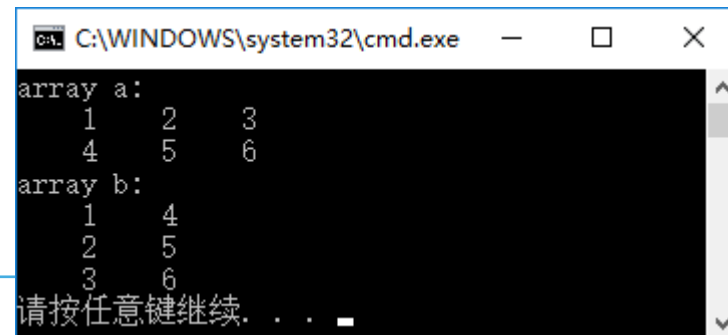
二维数组程序举例

【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。

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

```
#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数组中的一行中各元素
    {
        for (j=0;j<=2;j++)      //处理a数组中某一列中各元素
        {
            printf("%5d",a[i][j]); //输出a数组的一个元素
            b[j][i]=a[i][j]; //将a数组元素的值赋给b数组相应元素
        }
    }
```

```
        printf("\n");
    }
    printf("array b:\n");
    for(i=0;i<=2;i++)           //输出b数组各元素
                                //处理b数组中一行中各元素
    {
        for(j=0;j<=1;j++)      //处理b数组中一列中各元素
                                //输出b数组的一个元素
        {
            printf("%5d",b[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```



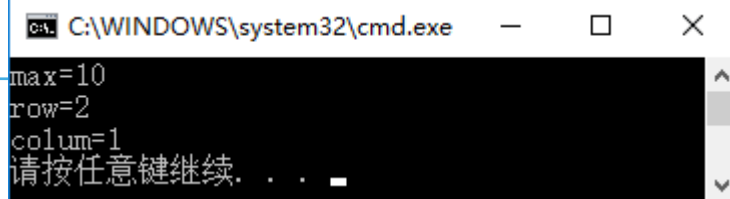
二维数组程序举例

算法

找最大最小值
打擂台算法

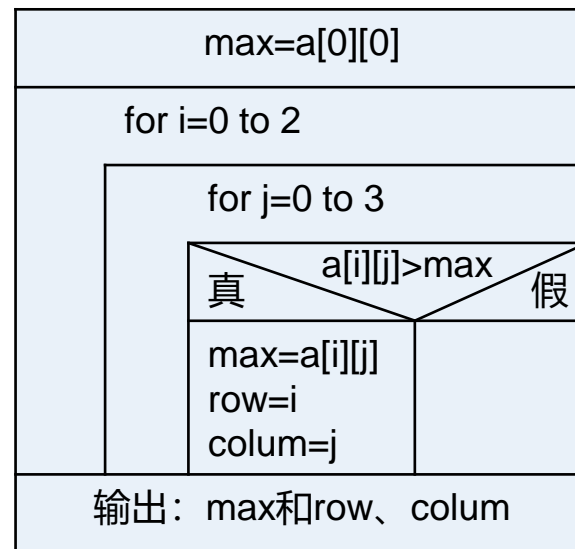
【例6.5】有一个 3×4 的矩阵，要求编程序求出其中值最大的那个元素的值，以及其所在的行号和列号。

```
#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}};    //定义数组并赋初值
    max=a[0][0];                                           //先认为a[0][0]最大
    for(i=0;i<=2;i++)
        for(j=0;j<=3;j++)
            if(a[i][j]>max)                                //如果某元素大于max, 就取代max的原值
            {   max=a[i][j];
                row=i;                                     //记下此元素的行号
                column=j;                                  //记下此元素的列号
            }
    printf("max=%d\nrow=%d\ncolumn=%d\n",max,row,column);
    return 0;
}
```



```
C:\WINDOWS\system32\cmd.exe
max=10
row=2
column=1
请按任意键继续. . .
```

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。以后每一个人都是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。



多维数组

```
float a[2, 3, 4];    //定义三维数组a, 可理解为2页, 每页有一个3行4列的二维数组
```

第1维的下标变化最慢, 最右边的下标变化最快。

在内存中的排列顺序为:

$a[0][0][0] \rightarrow a[0][0][1] \rightarrow a[0][0][2] \rightarrow a[0][0][3] \rightarrow a[0][1][0] \rightarrow a[0][1][1] \rightarrow a[0][1][2] \rightarrow a[0][1][3] \rightarrow a[0][2][0] \rightarrow a[0][2][1] \rightarrow a[0][2][2] \rightarrow a[0][2][3] \rightarrow$

$a[1][0][0] \rightarrow a[1][0][1] \rightarrow a[1][0][2] \rightarrow a[1][0][3] \rightarrow a[1][1][0] \rightarrow a[1][1][1] \rightarrow a[1][1][2] \rightarrow a[1][1][3] \rightarrow a[1][2][0] \rightarrow a[1][2][1] \rightarrow a[1][2][2] \rightarrow a[1][2][3]$



字符数组与字符串

定义字符数组

每个元素为一个字符。

```
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，因此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
```

字符数组的初始化

```
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
```

初值个数**大于数组长度**，则出现语法错误。

初值个数**小于数组长度**，则只将初始字符赋给前面的元素，其余元素自动赋值为**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，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[]={'l',' ','a','m',' ','h','a','p','p','y'};    //数组c的长度自动定为10
```

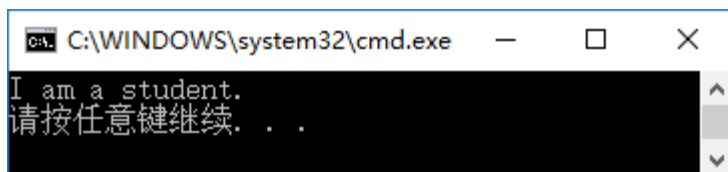
```
char diamond[5][5]={{' ',' ','*'},{' ','*',' ','*'},{'*',' ',' ','*'},{' ','*',' ','*'},{' ',' ','*'}};
```

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

引用字符数组中的元素

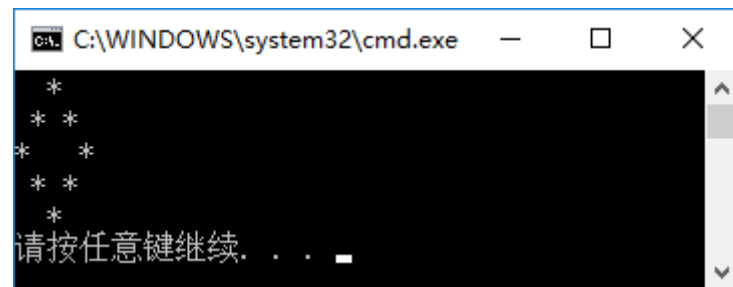
【例6.6】输出一个已知的字符串。

```
#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;
}
```

A screenshot of a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The output shows the string 'I am a student.' followed by a prompt '请按任意键继续. . .'.

【例6.7】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;
}
```

A screenshot of a Windows command prompt window titled 'C:\WINDOWS\system32\cmd.exe'. The output shows a diamond shape made of asterisks:
*
* *
* * *
* *
*
followed by a prompt '请按任意键继续. . .'.

```
int main()
```

```

{
    char a[7] = { ' ', ' ', ' ', ' ', ' ', ' ', ' ' };
    int i, n;
    for (i = 0; i < 4; i++)
    {
        printf("\n");
        a[3 + i] = a[3 - i] = '*';
        for (n = 0; n < 7; n++)
        {
            printf("%c", a[n]);
        }
    }
    for (i = 3; i >= 0; i--)
    {
        printf("\n");
        a[3 + i] = a[3 - i] = ' ';
        for (n = 0; n < 7; n++)
        {
            printf("%c", a[n]);
        }
    }
    return 0;
}

```

	0	1	2	3	4	5	6
i=0	spc	spc	spc	*	spc	spc	spc
i=1	spc	spc	*	*	*	spc	spc
i=2	spc	*	*	*	*	*	spc
i=3	*	*	*	*	*	*	*
i=3	spc	*	*	*	*	*	spc

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

C:\Users\86185\Desktop\115作业\Debug\1112作业.exe (进程 7500) 已退出。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->
“按任意键关闭此窗口”。...

字符串和字符串结束标志

在C语言中，字符串作为字符数组来处理的。

例如"C program"占10个字节，字符占9个字节，最后一个字节'\0'为系统自动加上

C语言规定以字符'\0'作为字符串结束标志。

注意

- 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。

字符串和字符串结束标志

0	1	2	3	4	5	6	7	8	9	10
I		a	m		h	a	p	p	y	\0

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

或 char c[]="I am happy";

用一个字符串(字符串的两端用双引号)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
---	---	---	---	---	----	----	----	----	----

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) **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。

```
#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;
}
```

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

-
- (1) 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'\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。
-



```
//字符串格式输入，数组名代
//表数组起始地址
```

China ✓

系统会自动在China后面加一个'\0'结束符。

如果利用一个scanf函数输入多个字符串，则应在输入时以**空格**分隔。

从键盘输入:

How are you? ↙

由于有空格字符分隔，作为3个字符串输入。

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

从键盘输入:

How are you? ↙

由于系统把空格字符作为输入的字符串之间的分隔符，因此只将空格前的字符“How”送到str中。

[illegible]

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("%x",c); //用16进制形式输出数组c的起始地址
```

c数组

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

- 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。

```
printf("%s",c); //输出字符串
```



使用字符串处理函数

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


输出字符串的函数

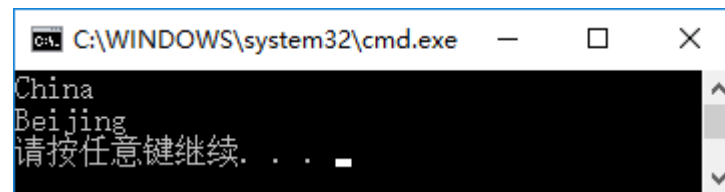
puts(字符数组)

作用：将一个以'\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。

输出完字符串后换行。

```
#include <stdio.h>
int main()
{
    char str[]={"China\nBeijing"};
    puts(str);
    return 0;
}
```





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，并且得到字符数组的起始地址。

C 库函数 `char *gets(char *str)` 从标准输入读取一行，并把它存储在参数 `str` 所指向的字符串中。当读取到换行符时，它会停止。

`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" 和 `\0` 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` 函数只能输出或输入一个字符串。

2011年12月，ANSI 采纳了 ISO/IEC 9899:2011 标准，标准中删除了 `gets()` 函数，使用一个新的更安全的函数 `gets_s()` 替代，`gets_s(char *buff, size)`

字符串连接函数

strcat(字符串1, 字符串2)

char * strcat(char *dest, const char *src)

作用：把字符串2接到字符串1的后面，结果放在字符串1中

函数返回值为——字符串1的地址。

字符串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

	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
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	\0																								
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

字符串复制函数

strcpy(字符数组1, 字符串2)

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

执行后, str1: C h i n a \0 \0 \0 \0 \0

`char *strcpy(char *dest, const char *src)`

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的长度大于等于字符串2的长度。

strncpy(字符数组1, 字符串2)

`char *strncpy(char *dest, const char *src, size_t n)`

字符串2中前面n个字符复制到字符数组1中去

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;
```



```

1  #define _CRT_SECURE_NO_WARNINGS
2  #include<stdio.h>
3  #include<math.h>
4  #include<string.h>
5
6  int main()
7  {
8      char src[50], dst[50];
9      strcpy(src, "This is source");
10     strcpy(dst, "This is destination");
11
12     strcat(dst, src);
13     printf("最终目标字符串为: %s", dst);
14     return 0;
15 }
16

```

Microsoft Visual Studio 调试控制台

最终目标字符串为: This is destinationThis is source

字符串比较函数

`strcmp(字符串1, 字符串2)`

```
int strcmp(const char *str1, const char *str2)
```

作用：比较字符串1和字符串2。

两个字符数组的第一个元素开始，顺序比较数组中各个元素的ASCII码值，返回结果：

- 如果返回值小于 0，则表示 str1 小于 str2。
- 如果返回值大于 0，则表示 str1 大于 str2。
- 如果返回值等于 0，则表示 str1 等于 str2。

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

```
int main()  
{  
    char str1[15];  
    char str2[15];  
    int rst;  
  
    strcpy(str1, "aacde");  
    strcpy(str2, "aACDE");  
  
    rst = strcmp(str1, str2);  
    if (rst < 0)  
        printf("str1 小于 str2\n");  
    else if (rst > 0)  
        printf("str1 大于 str2\n");  
    else  
        printf("str1 等于 str2\n");  
  
    return 0;  
}
```

Microsoft Visual Studio 调试控制台

str1 大于 str2





字符数组应用举例

【例6.9】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	\0
str[1]:	C	h	i	n	a	\0	\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	\0

char str[3][20]; //定义二维字符数组

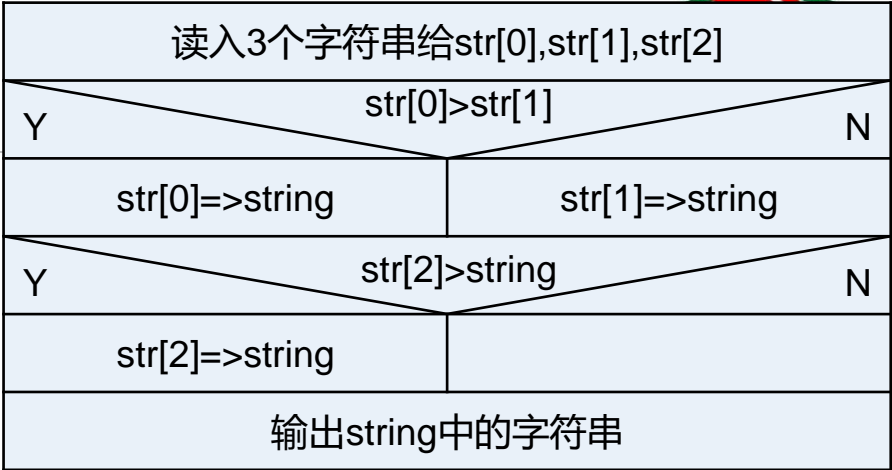
```
C:\WINDOWS\system32\cmd.exe
Holland
China
America

the largest string is:
Holland
请按任意键继续. . .
```

字符数组应用举例

【例6.9】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	\0
str[1]:	C	h	i	n	a	\0	\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	\0

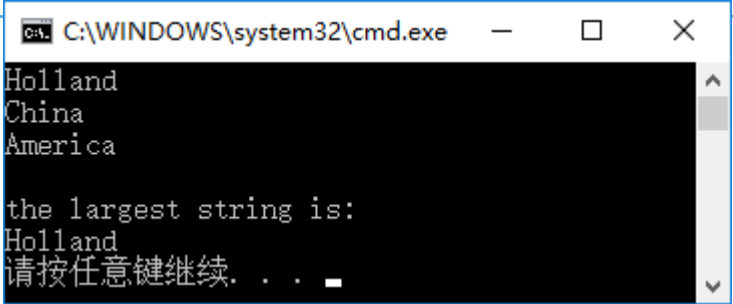


```
#include<stdio.h>
#include<string.h>
int main()
{
    char str[3][20];    //定义二维字符数组
    char string[20];
    //定义一维字符数组，作为交换字符串时的临时字符数组
    int i;
    for(i=0;i<3;i++)
        gets(str[i]);    //读入3个字符串，分别给str[0],str[1],str[2]
```

```
    if(strcmp(str[0],str[1])>0)    //若str[0]大于str[1]
        strcpy(string,str[0]);    //把str[0]的字符串赋给字符数组string
    else                            //若str[0]小于等于str[1]
        strcpy(string,str[1]);    //把str[1]的字符串赋给字符数组string
    if(strcmp(str[2],string)>0)    //若str[2]大于string
        strcpy(string,str[2]);    //把str[2]的字符串赋给字符数组string
    printf("\nthe largest string is:\n%s\n",string);    //输出string
    return 0;
}
```



- (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。



小结

puts(字符数组)

gets(字符数组)

strcat(字符数组1, 字符数组2)

strcpy(字符数组1, 字符串2)

strncpy(字符数组1, 字符串2)

strcmp(字符串1, 字符串2)

strlen(字符数组)

strlwr(字符串)

strupr(字符串)

注意

- 以上介绍了常用的字符串处理函数，它们是C语言编译系统提供的公共函数，C库函数
- 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，必要时查询库函数手册。
- `#include <string.h>`

<https://www.runoob.com/cprogramming/c-function-strcmp.html>