



福州大学至诚学院  
FUZHOU UNIVERSITY ZHICHENG COLLEGE

# 高级语言程序设计 (C语言与数据结构)

杨雄

83789047@qq.com



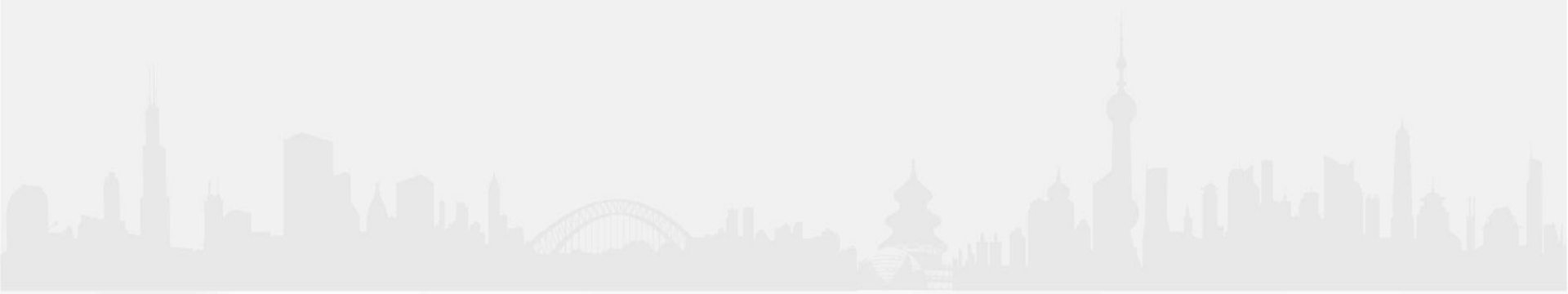


# 第六章 数组

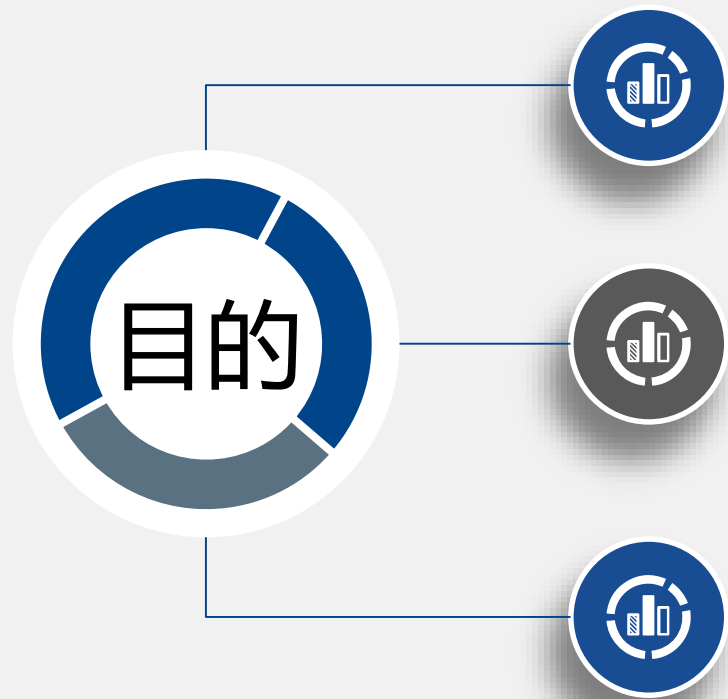
## 6.1 一维数组

## 6.2 二维数组

## 6.3 字符数组



# 学习目的



掌握数组的概念及数组的定义

掌握在程序设计中的  
如何使用数组


字符数组是C语言存放字符串  
的主要方法，  
并注意字符串结束标志的规定




Part.1

## 6.1 一维数组

## 6.1 一维数组

 前面介绍的数据类型（如整型，字符型，实型）  
是C语言的基本数据类型。基本类型变量称简单  
变量。

 每个简单变量在内存中都占据独立的存储单元，  
**相互之间没有直接的联系。**

## 6.1 一维数组

假如我们要求**100**个人工资的总和，如果把每个人的工资分别用变量**S1, S2, ..., S100**表示，那么在程序中就要定义**100**个变量。

计算表达式为：

$$\text{sum} = s1 + s2 + \dots + s100$$

显然，定义的变量过多、程序繁琐冗长，可读性差。



## 6.1 一维数组

使用数组可把每个人的工资用数组元素来表示。

$s[0], s[1], s[2], \dots, s[99]$

这样，在程序中只需定义一个具有100个数组元素的数组s。  
计算工资总和可使用以下结构：

```
for( i=0; i<=99; i++)
```

```
    sum=sum+s[i];
```

使用数组可以使程序结构更加合理、简洁、清晰，提高程序的功能和灵活性。

## 6.1.1 一维数组定义

· 格式:

**类型说明符** **数组名** [**常量表达式**]

例: `int a[10]`

`a[0], a[1], a[2], a[3], ..., a[9]`

- 数组是数目固定, 类型相同的若干变量的有序集合。
- 数组中的每一个变量称为数组元素, 属于同一个数据类型。
- 数组在内存中占有一段连续的存储空间。



## 6.1.1 一维数组定义

### 说明:

- 一个数组名在程序中只能说明一次, 不能重复说明。

如:            `int a[10] ;`  
                `float a[5] ;` ×

- 用方括号将常量表达式括起, 不能使用圆括号。

如:            `int a(5);` ×

- 常量表达式定义了数组元素的个数。

## 6.1.1 一维数组定义

类型说明符 数组名 [常量表达式];

- 常量表达式可以是**整数常量**、**符号常量或常量表达式**，不能包含**变量**，其值必须是**正整数**。

如: #define N 10

```
int a[12], b[N], c[ '0' ]; float d[2*N];
```

```
int n, m=5, d[m]; ×
```

```
scanf("%d", &n );
```

```
int c[n]; ×
```

```
int num[-8] ; ×
```

```
int m(8) ; ×
```

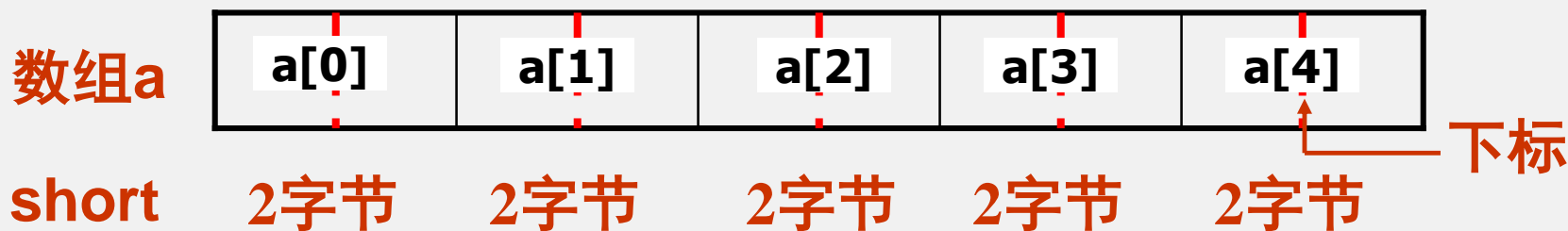
## 6.1.1 一维数组定义

一维数组在内存中的存放方式：

📖 数组定义以后，编译器就会为这个数组在内存中分配一串**连续的存储单元**用于存放数组元素的值。**数组名**表示存储单元的**首地址**，存储单元的多少由**数组的类型和数组的大小**决定。

如：

```
short int a[5];
```



一维数组的总字节数可按下式计算：

$\text{sizeof(类型)} \times \text{数组长度} = \text{总字节数}$     `printf("%d", sizeof(a));`

## 6.1.2 一维数组的引用

- 数组元素都是**变量**
- 只能逐个引用数组中的数组元素, 不能一次引用整个数组。
- 引用方式:    **数组名 [ 下标 ]**

如:

```
int  c[10];  
c[0] = 3;  
scanf("%d", &c[1]);  
printf("%d %d", c[0], c[1]);
```

## 6.1.2 一维数组的引用

说明： 数组名[**下标**]

- 下标可以是整数、符号常数、**变量或整型表达式**。

如：     **int c[10], x=3;**

**c[5-2] == c[3] == c[x]**

- 下界 $\leq$ 下标 $\leq$  上界

下界=0, 上界=整常量表达式-1。

- 对于长度为**n**的数组, 下标范围为**0~(n-1)**。

如：     **float ia[5] ; ia[5]=4. ;**     **×**

## 6.1.2 一维数组的引用

一维数组元素引用的**规定**：

- 数组必须**先定义后使用**
- 对数组中所有元素逐个引用时，通常使用**循环结构**。

例：        `int i, a[10];`

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

`a[i]=i;`



## 6.1.2 一维数组的引用

数组元素的**赋值**:

1. 用**赋值语句**给数组元素赋值
2. 用**输入函数**给数组元素赋值
3. **定义**数组时对数组元素赋初值

例:

```
float b[12];
```

```
b[1]=1.0;   b[3]=7.6;
```

```
b[0]=b[1]+b[3];
```

.....

## 6.1.2 一维数组的引用

### 例1 数组的输入与输出

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

```
void main( )
```

```
{ int i, a[5] ;
```

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

```
        scanf( “%d” , &a[i] ) ;
```

```
    for ( i =4; i>=0; i-- )
```

```
        printf( “a[%d]=%d ” , i, a[i] ) ;
```

```
}
```

运行:

**1 2 3 4 5** ✓

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

## 6.1.4 一维数组初始化

- 数组的初始化:

是**定义数组**时(在程序编译期间)完成赋初值任务。

- 格式:

**类型符 数组名[表达式] = {初值表};**

## 6.1.4 一维数组初始化

- 给全部元素赋初值

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

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

—若初始数据个数 > 数组长度, 编译出错。

- 给部分元素赋初值

例: `int a[8]={ 0, 1, 2, 3, 4 };`

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

—剩下的元素的初值是0

## 6.1.4 一维数组初始化

- 给**全部**元素赋初值时可以**不指定数组的长度**

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

— 数组的长度就是初值表中数值的个数

- 注意: 如果数组长度与初始数据个数不相等, 在定义数组时**不能省略**数组长度。
- 当对全部数组元素初始化为 **0** 时, 可以写成:

`int x[5] = {0, 0, 0, 0, 0};`

或: `int x[5] = {0};`

## 6.1.4 一维数组初始化

说明:

- 初值表不能为空

```
int a[5] = { }; ×
```

- 没有初始化的数组，其元素的值不确定。

```
int a[5]={ 1 },i;  
for ( i=0; i<5; i++)  
printf( “ %d” ,a[i]);
```

1 0 0 0 0

```
int a[5],i;  
for ( i=0; i<5; i++)  
printf( “ %d” ,a[i]);
```

872 0 1492 4160 186

不可预知



## 6.1.4 一维数组初始化

例 指出下面定义的数组，哪些是错误的 b d e。

```
#define S 20
```

a) `int i[4*10] , k[20];`

b) `int j=4;`  
`float x[2*j-1];`

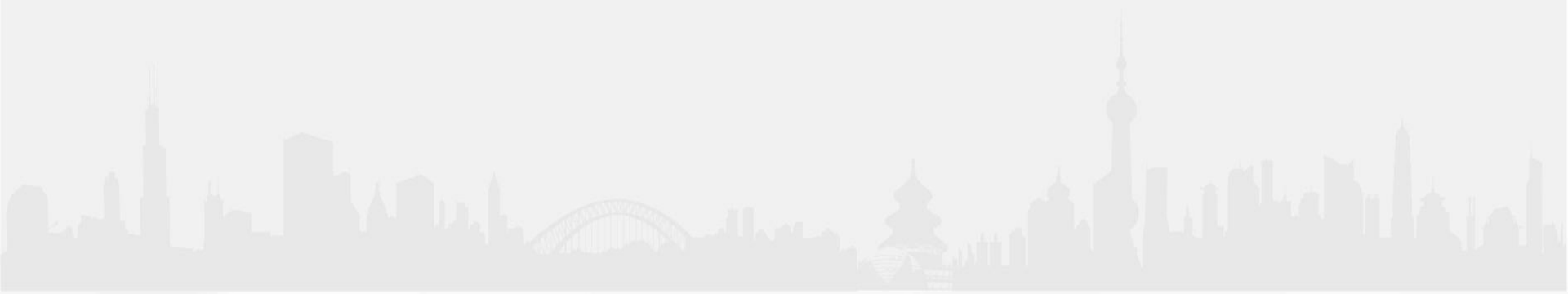
c) `int m[S];`

d) `int n[3] = {1, 2, 3, 4};`

e) `int h[S-25] ;`

# Part.2

## 6.2 二维数组



## 6.2.1 二维数组定义

- 二维数组是由具有**两个下标**的数组元素组成。

格式:

**类型符 数组名 [常量表达式1] [常量表达式2];**

如: `int a[3][4];`

	0列	1列	2列	3列	
0行	a[0][0]	a[0][1]	a[0][2]	a[0][3]	数组元素a[1][1] ↓ ↓ 行 列
1行	a[1][0]	a[1][1]	a[1][2]	a[1][3]	
2行	a[2][0]	a[2][1]	a[2][2]	a[2][3]	

## 6.2.1 二维数组定义

例:

```
#define N 3
```

```
#define M N+2
```

```
float s[M][N];
```

即定义了一个二维浮点型数组s, 数组s的元素个数和最后一个元素分别为 15、s[4][2]。

## 6.2.1 二维数组定义

二维数组的存储方式:

- 二维数组的元素是按**行顺序**存放的。

如: `int a[3][4];`

- 二维数组的总字节数:

行数 × 列数 × 类型字节数

= 总字节数

`printf(" %d", sizeof(a));`



## 6.2.2 二维数组引用

- 引用格式:

数组名[行下标][列下标]

```
int a[3][4];  
for (m=0; m<3; m++)  
    for (n=0; n<4; n++)  
        printf( "%d" , a[m][n] );
```



## 6.2.3 二维数组初始化

- 二维数组的初始化通常是按**行**进行赋值的

— 格式:

**类型符 数组名[表达式1][表达式2]={初值表};**

— 三种方式:

- ★ 给全部元素赋初值
- ★ 给部分元素赋初值
- ★ 给**全部**元素赋初值时, **不指定第一维**的长度, 但要**指定第二维**的长度。

## 6.2.3 二维数组初始化

- 给全部元素赋初值

用括号按行分组

如: `int x[2][4]={{1, 2, 3, 4}, {6, 7, 8, 9}};`

或写成:

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

数组x中各元素的值为:

x[0][0]	x[0][1]	x[0][2]	x[0][3]
1	2	3	4
x[1][0]	x[1][1]	x[1][2]	x[1][3]
6	7	8	9

## 6.2.3 二维数组初始化

- 给部分元素赋初值

没有明确初始化的  
元素被自动初始化

如: `int a[3][4]={1, 2, 3, 4, 5};`

1	2	3	4
5	0	0	0
0	0	0	0

可用分行赋值的方法对某行中部分元素赋初值

如: `int a[3][4]={{1}, {5}, {6}};`

1	0	0	0
5	0	0	0
6	0	0	0

## 6.2.3 二维数组初始化

- 对**全部**元素赋初值, 可以不指定第一维的长度, 但要指定第二维的长度。

```
int a[2][3] = {{10, 11, 12}, {13, 14, 15}};
```

```
int a[2][3] = {10, 11, 12, 13, 14, 15};
```

省略第一维的长度时, 也可对**部分**元素赋初值, 但应**分行**赋值。

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

1	2	3	0
0	0	0	0
0	10	0	0

## 6.2.3 二维数组初始化

例: 阅读下列程序

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

```
void main()
```

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

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

```
    n[i] = 0 ;
```

```
  k=2 ;
```

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

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

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

```
  printf( "%d \n" , n[k] ) ;
```

```
}
```

输出结果:

✓ A> 不确定的值

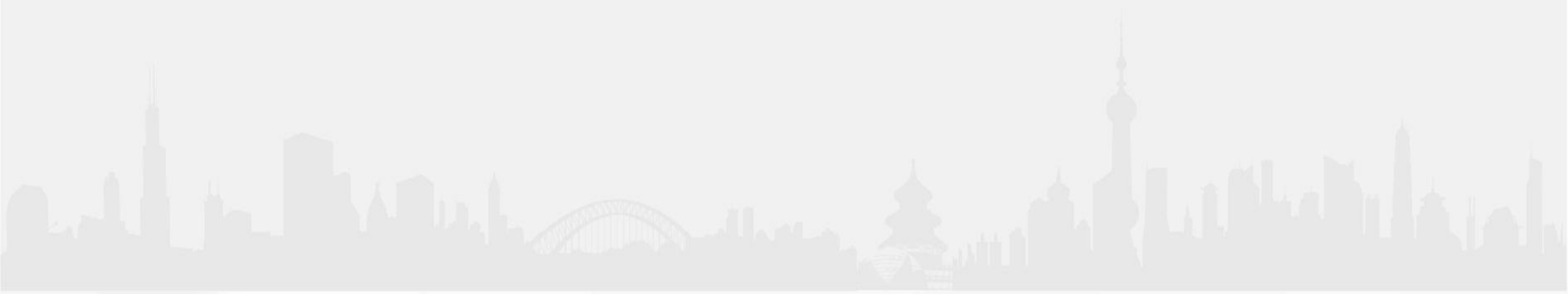
B> 3

C> 2

D> 1

# Part.3

## 6.3 字符数组





## 6.3 字符数组

6.3.1 字符数组的定义、初始化

6.3.2 字符数组的输入与输出

6.3.3 字符串处理函数

6.3.4 字符数组应用实例

## 6.3.1 字符数组定义、初始化

- 用来存放**字符**型数据的数组。在字符数组中，每一个数组元素只能存放**一个**字符。
- 格式：

**char 数组名[常量表达式];**

**char 数组名[常量表达式1][常量表达式2];**

如: **char a[6], b[5][10];**

## 6.3.1 字符数组定义、初始化

### • 用字符常数初始化数组

例:

```
char c[8]={ 'c', 'h', 'i', 'n', 'a'};
```

- 若字符个数 < 数组长度, 则将这些字符赋给前面的元素, 其余元素自动为空字符( '\0' )。
- 若字符个数 > 数组长度, 则作错误处理。

存储形式:

c[0]	c[1]	c[2]	c[3]	c[4]	c[5]	c[6]	c[7]
'c'	'h'	'i'	'n'	'a'	'\0'	'\0'	'\0'

## 6.3.1 字符数组定义、初始化

- 用字符串为字符数组赋初值

例:

```
char a[10]={ "china" };
```

或:

```
char a[6]= "china" ;
```

**C语言中所有字符串都是以' \0' 结束。**

存储形式为:

c	h	i	n	a	\0
---	---	---	---	---	----

来自字符串常量的  
结束字符 '\0' 。

## 6.3.1 字符数组定义、初始化

- 用**字符串**为字符数组赋初值比用**字符常数**赋值时**要多占一个字节**。

用字符初始化时,不要求最后一个字符一定为  
'\0' 。

```
char c[5]={ 'C', 'h', 'i', 'n', 'a'};
```

```
char a[6]= "China" ;
```

## 6.3.1 字符数组定义、初始化

例:

```
char s1[10]={ 'C' , '␣' , 'P' ,  
             'r' , 'o' , 'g' , 'r' , 'a' , 'm' };
```

```
char s2[10]={ "C␣Program" };
```

```
char s3[10]= "C␣Program" ;
```

```
char s4[ ]= "C␣Program" ;
```

存储形式为:

C	␣	P	r	o	g	r	a	m	\0
---	---	---	---	---	---	---	---	---	----

## 6.3.2 字符数组输入、输出

- 有以下方式:

- 逐个字符输入输出 `%c`

- 作为整体一次输入输出 `%s`

- 用字符串输入输出函数

- ✧ `gets()`

- ✧ `puts()`

## 6.3.2 字符数组输入、输出

1. 用**%c** 对字符数组元素逐个输入、输出字符。

例: `#include <stdio.h>`

`void main( )`

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

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

`scanf("%c", &a[i])`  $\Rightarrow$  `a[i]=getchar( );`

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

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

`printf("\n");`

`}`



## 6.3.2 字符数组输入、输出

2. 用**%s**, 对字符数组整体输入或输出字符串。

例:

```
#include <stdio.h>
void main( )
{ char a[6] ;
  scanf( "%s" , a);
  printf( "%s" , a);
  printf( "\\n" );
}
```

运行情况如下:

输入: **China**✓

输出结果: **China**

## 6.3.2 字符数组输入、输出

注意：

- scanf函数参数要求的是**地址**，故直接用**字符数组名**进行操作。
- 输出字符**不包括结束符** ‘\0’ 。
- 用**%s**格式符输出字符串时，输出项**只能是字符数组名**，不能是数组元素名。

printf( "%s\n" , a[0]); ×

## 6.3.2 字符数组输入、输出

注意:

- `%s` 输入字符串时, 遇**空格、回车符、Tab**结束输入。  
不能接收空格。

若 `char c1[6], c2[6];`  
`scanf( "%s%s" , c1, c2 );`

输入:

`a_good_book` ✓

`printf( "%s %s\n", c1, c2 );`

输出: `a good`

c1	a	\0	任意	任意	任意	任意
c2	g	o	o	d	\0	任意

## 6.3.2 字符数组输入、输出

注意：

- 若一个字符数组中含有一个或多个 “\0”，则遇到第一个 “\0” 时结束输出。

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

```
void main()
```

```
{ char a[10]={ “Boy\0abc” };
```

```
    printf( “%s” , a );
```

```
    printf( “\n” );
```

```
}
```

输出结果： Boy

## 6.3.2 字符数组输入、输出

### 3. 利用字符串输入和输出函数

- 字符串输入函数

- 格式:

**gets(字符数组名);**

- 作用: 将输入的字符串赋给字符数组。输入时, 遇第一个回车符结束输入。可接收**空格**、**制表符**。

- gets( )函数同scanf( )函数一样, 在读入一个字符串后, 系统自动在字符串后加上一个字符串结束标志' \0' 。

- 函数gets( )只能一次输入一个字符串。

## 6.3.2 字符数组输入、输出

例：函数gets()与scanf()的区别

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

```
void main()
```

```
{ char str1[20], str2[20];
```

```
    gets(str1);
```

```
    scanf( "%s" , str2);
```

```
    printf( "str1: %s\n" , str1);
```

```
    printf( "str2: %s\n" , str2);
```

```
}
```

输入： program C✓  
          program C✓

输出： str1: program C  
          str2: program

## 6.3.2 字符数组输入、输出

### ●字符串输出函数

#### – 格式:

**puts(字符数组名); 或 puts(字符串);**

– 作用: 输出字符数组的值, 遇' \0' 结束输出。

– 注意: puts()一次只能**输出一个字符串**。**输出字符串后自动换行**。可以输出转义字符。

– printf()函数可以**同时输出多个字符串**, 并且能灵活控制是否换行, 所以printf()函数比puts()函数更为常用。

## 6.3.2 字符数组输入、输出

例：函数puts()与printf()的区别

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

```
void main()
```

```
{ char str1[] = "student", str2[] = "teacher" ;
```

```
    puts(str1);
```

```
    puts(str2);
```

```
    printf( "%s" , str1);
```

```
    printf( "%s\n%s" , str1, str2 );
```

```
}
```

输出结果：

**student**

**teacher**

**studentstudent**

**teacher**



## 6.3.2 字符数组输入、输出

例1: 指出下面数组引用哪些是错误的, 为什么?

**char str[8] , c ;**

- (1) **str[1] = '3' ;**
- (2) **scanf( "%s" , str) ;**
- (3) **str = "program" ;** ×
- (4) **str = getchar( ) ;** ×
- (5) **c = str + '3' ;** ×
- (6) **str[8] = 'a' ;** ×

## 6.3.2 字符数组输入、输出

**例2: 下面是对数组进行初始化的语句, 指出哪些是错误的?**

- (1) `int m[ ]={1, 2, 3, 4, 5};`**
- (2) `char str[2][ ]={ “boys” , “girls” };` ✕**
- (3) `int n[ ][ ]={{1, 2, 3},{5, 6, 7}};` ✕**
- (4) `char str[20]={ “I am a student” };`**

## 6.3.3 字符串处理函数

**#include "string.h"**

**strcat(字符数组1, 字符数组2) ;**

**strcpy(字符数组1, 字符串2) ;**

**strlen(字符串) ;**

**strlwr(字符串) ;**

**strupr(字符串) ;**

**strcmp(字符串1, 字符串2) ;**

## 6.3.3 字符串处理函数

### ●字符串连接函数

输出：  
**Hello world**

–格式: **strcat**( 字符串数组1, 字符串数组2 );

–作用: **连接**两个字符串数组中的字符串, 将字符串2连接到字符串数组1的后面, 结果放在字符串数组1中。

–例: `char a[15] = "Hello";`  
`strcat( a, "_world" );`  
`printf( "%s" , a);`

字符串数组1的' \0' 将被字符串2覆盖, 连接后生成的新的字符串的最后保留一个' \0' 。

a	H	e	l	l	o	\0	\0	\0	\0	\0	\0	\0	\0	\0
	_	w	o	r	d	\0								
a	H	e	l	l	o	_	w	o	r	d	\0	\0	\0	\0

## 6.3.3 字符串处理函数

### ●字符串拷贝函数

- 格式: **strcpy**( 字符数组1, 字符串2 );
- 作用: 将**字符串2**拷贝到**字符数组1**中。只复制第一个' \0' 前的内容(含' \0' )。
- 例: 

```
char a[20] = "Hello";  
strcpy( a, "world" );  
printf( "%s" , a);
```

**输出:**      world

## 6.3.3 字符串处理函数

### ● 字符串长度函数

- 格式: `strlen(字符串)` ;
- 作用: 返回字符串中有效字符的个数, **不包括结束符 `'\0'`** 。
- 例1:  

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

输出 8

### 6.3.3 字符串处理函数

例2: `char sp[10] = "\\t\\v\\\\0will\\n" ;`  
`printf( "%d\\n" , strlen(sp));`

A)14

B)✓3

C)9

D)输出值不定

例3: `char st[11] = "\\x69\\082\\n" ;`  
`printf( "%d\\n" , strlen(st));`

A)3

B)5

✓C)1

D)输出值不定

### 6.3.3 字符串处理函数

例4: `char st[ ]={ "hello\0\t\nabc" };`  
`printf( "%d\n" , sizeof(st));` 数组st的长度是

- A)10      B)8      C)6      ✓D)12

例5: `char a[ ]={ "I\nsee\ " ABC\ "" };`  
数组a的长度是

- A)7      B)8      C)9      ✓D)11



## 6.3.3 字符串处理函数

例: strlen()的使用

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

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

```
void main()
```

```
{ int i; char str1[20], str2[20];
```

```
  gets(str1); gets(str2);
```

```
  for(i=0; str2[i] != '\0' ; i++);
```

```
  printf( "%s : %d\n" , str1, strlen(str1));
```

```
  printf( "%s : %d\n" , str2, i );
```

```
  printf( "%s : %d\n" , "Good luck" , strlen( "Good luck" ));
```

} 这两句的功能等同于函数strlen(), i 返回串长。

输入: Good luck✓  
Good luck✓

输出: Good luck: 9  
Good luck: 9  
Good luck: 9

## 6.3.3 字符串处理函数

### ●字符串比较函数

– 格式: `strcmp( 字符串1, 字符串2 );`

– 作用: 比较字符串1和字符串2

```
printf(“%d\n”,strcmp(“b”,“b”));
```

输出0

```
printf(“%d\n”,strcmp(“b”,“f”));
```

输出-1

✧ 字符串1 == 字符串2, 函数返回值为0。

✧ 字符串1 > 字符串2, 函数返回值为 1。

✧ 字符串1 < 字符串2, 函数返回值为-1。

## 6.3.3 字符串处理函数

- **注意:**

**对两个字符串比较，不能用以下形式:**

```
if( str1 == str2 ) printf( "yes" ); ×
```

**而只能用:**

```
if( strcmp(str1, str2) == 0) printf( "yes" );
```

## 6.3.3 字符串处理函数

- **strlwr(字符数组名);**

**作用：将字符数组中大写字母转换成小写字母**

- **strupr(字符数组名);**

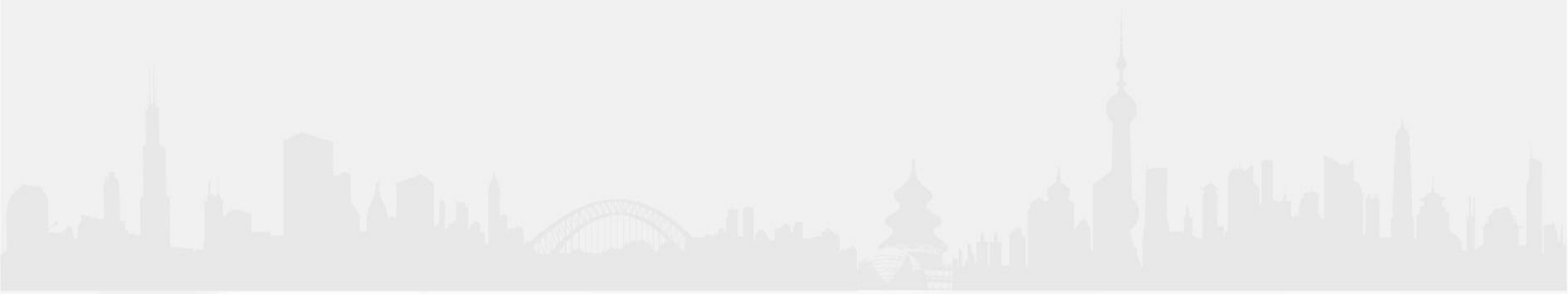
**作用：将字符数组中小写字母转换成大写字母**

**例：** `char a[20] = "Hello" ;`  
`printf( "%s, %s" , strlwr(a), strupr(a) );`

**输出： hello, HELLO**

# Part.4

## 总结



# 总 结

- 1.什么是数组？为什么要使用数组？如何定义数组？如何引用数组元素？
- 2.数组元素在内存中按什么方式存放？
- 3.什么是字符串？字符串结束符的作用是什么？
- 4.如何实现字符串的存储和操作，包括字符串的输入和输出？
- 5.怎样理解C语言将字符串作为一个特殊的一维字符数组？

## 习题

1. 若定义数组并初始化 `int a[10]={ 1,2,3,4 };`  
以下语句哪一个成立？
- A) 若引用`a[10]`，编译时报错。
  - B) 若引用`a[10]`，连接时报错。
  - ✓ C) 若引用`a[10]`，运行时出错。
  - D) 若引用`a[10]`，系统不报错。

## 习 题

2. 设有: `int x[2][4]={1, 2, 3, 4, 5, 6, 7, 8};`  
`printf( "%d" , x[2][4]);`

- A) 8      B) 1      C) ☒ 随机数      D) 语法检查出错

3. 设有: `int a[ ][3]={{1,2,3},{4,5,6},{7,8,9}};`  
则 `a[1][2]` 的初值为:

- A) 2      B) 4      C) ☒ 6      D) 8



## 4. 合法的数组定义是:

- A) `int a[ ] = "string" ;`
- B) `int a[5]={0,1,2,3,4,5};`
- C) `char a= "string" ;`
- ✓ D) `char a[ ]={ '0' , '1' , '2' , '3' };`

## 习 题

### 5. 阅读程序:

```
void main( )  
{ char a[10];  int i;  
  for( i=1; i<=5; i++)  
    scanf( "%c" , &a[i]);  
  printf( "%c" , a[0]);  
}
```

设从键盘输入字符串: **abcde**, 则程序的输出结果是:

A) a

B)空格

C)不确定

D)0

6. 以下定义语句中，错误的是：

A) `int a[ ] = {6,7,8};`

B) `int n=5, a[n];` ×

C) `char a[ ] = "string" ;`

D) `char a[5]={ '0' , '1' , '2' , '3' , '4' };`

## 7. 以下数组定义中正确的是:

- A) `float f[3,4];`
- B) `int a[ ][4];`
- C) `char c(3);`
- ✓ D) `double d[3+2][4];`

## 8. 若有以下的定义和语句:

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

```
char c= 'e' ;
```

则数值为2的表达式是

A) str[ 'g' -c]

B) str[2]

✓ C) str[ 'd' - 'c' ]

D) str[ 'd' -c]

## 习题

9. 若有数组定义 `char str[10]={ "1234567" };`  
`printf( "%.3s" , str);`

输出结果是:

- ✓ A) 输出第1, 第2, 第3个元素的值。
- B) 输出`str[1]`, `str[2]`, `str[3]`的值。
- C) 输出全部元素
- D) 输出格式描述不正确

## 10. 以下程序的输出结果是

```
void main()  
{ char  ch[3][5]={ "AAAA" , "BBB" ,  
  "CC" };  
  printf( "\ " "%s\ " \n" , ch[1]);  
}
```

- A) "AAAA"      B) "BBB" ✓  
C) "BBBCC"     D) "CC"