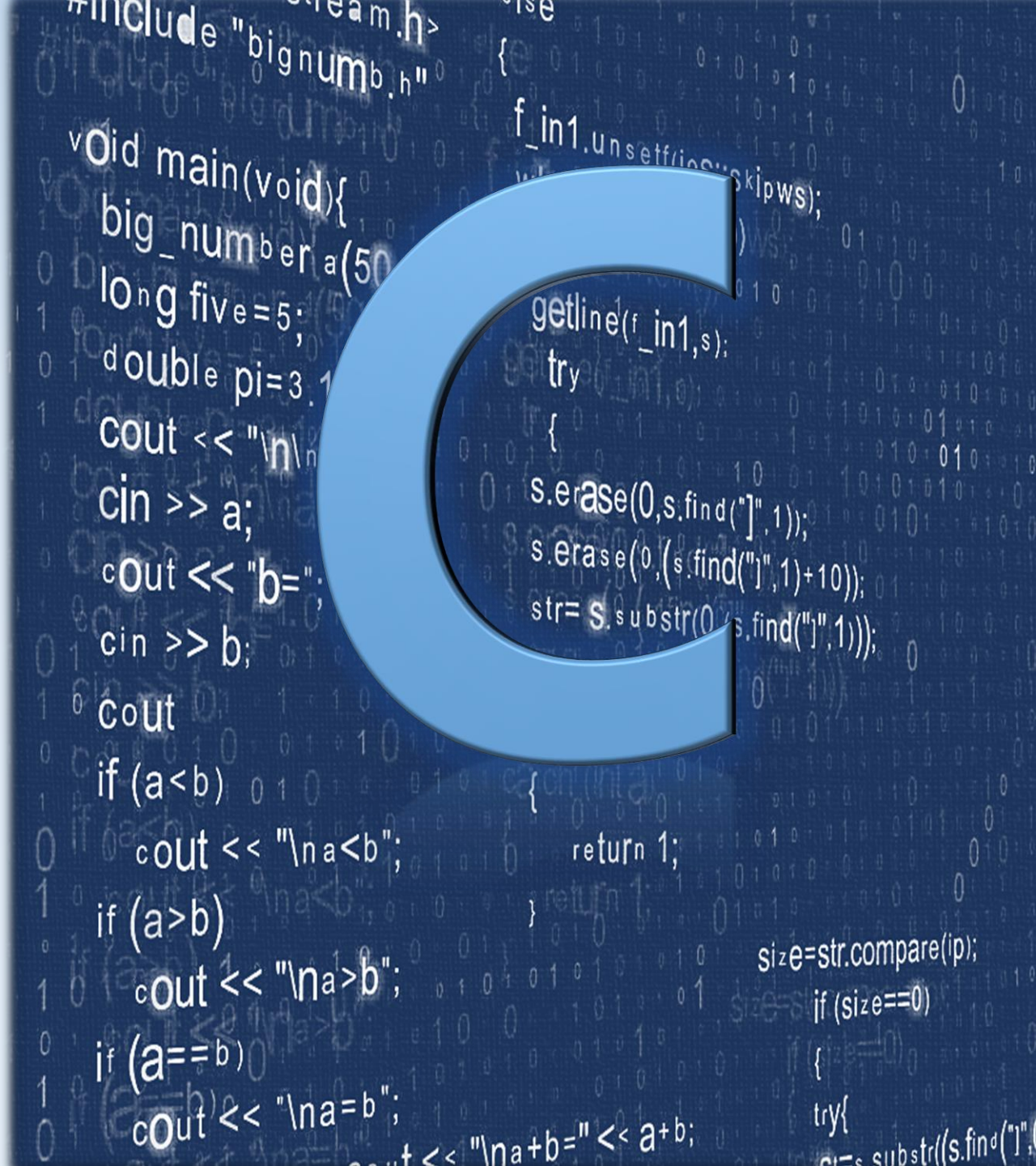


# 《C语言程序设计》

C语言课程组




# 上一讲知识复习

- ◆掌握三种基本结构的控制流程。
- ◆熟练掌握C语言的语句：基本语句、分支语句（条件语句）、循环语句。
- ◆着重掌握分支、多重循环的执行过程。
- ◆能读懂程序，明白该程序功能。

# 本讲教学目标

- ◆掌握数组声明的方法。
- ◆掌握一维数组、二维数组在内存中的存储。
- ◆掌握通过下标方式访问数组中各元素的方法。

# 本章授课内容



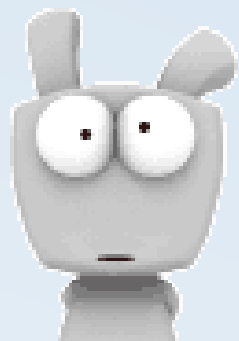
一维数组

二维数组

一维字符数组与字符串

# 一维数组

❖ 保存某班级所有学生的C语言成绩，并将成绩排序后打印输出



问题：  
如何保存全  
班成绩？

```
int a,b,c,d,e,f..... ?
```

# 一维数组

❖ 数组：类型相同的数据元素的集合。

◆ 这个类型可以是基本的数据类型，也可以是“构造类型”。

◆ 这些数据元素只能顺序的存放在内存的某段区域。

❖ 格式：**类型 数组名 [数组大小];**

◆ `int a[10];`



◆ `float c[20];`



◆ `char b[8];`



◆ `int d[4.0];`



# 一维数组

- ❖ 定义一个数组就等于同时定义多个变量。这些元素（变量）的名字：

**数组名[下标];** // 下标就是一个整数

- ❖ 例如:

**int a[10];** 这一行代码声明了一个10个元素的一维整型数组。

相当于声明了10个变量,变量名分别为

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

**注意:下标从0开始,共10个变量,最大下标为9**




# 一维数组

❖初始化方法一：全部初始化

```
int arr[5] = {11, 12, 13, 14, 15};
```

注意：

1、初值的个数等于数组的长度

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

2、每个初值的类型最好与数组的类型一致



# 一维数组

❖初始化方法二：部分元素赋初值

```
int b[5]={1,2};
```

$b[0]=1$     $b[1]=2$

$b[2] \sim b[4]$ 都为0

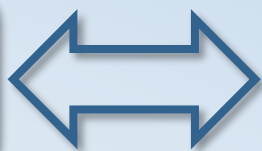
**注意：**

未赋值的均为0

# 一维数组

❖初始化方法三：省略长度赋初值

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



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

注意：

若被定义数组长度与提供初值的个数不相同，则不能省略长度定义数组

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

# 一维数组

## ❖ 练习

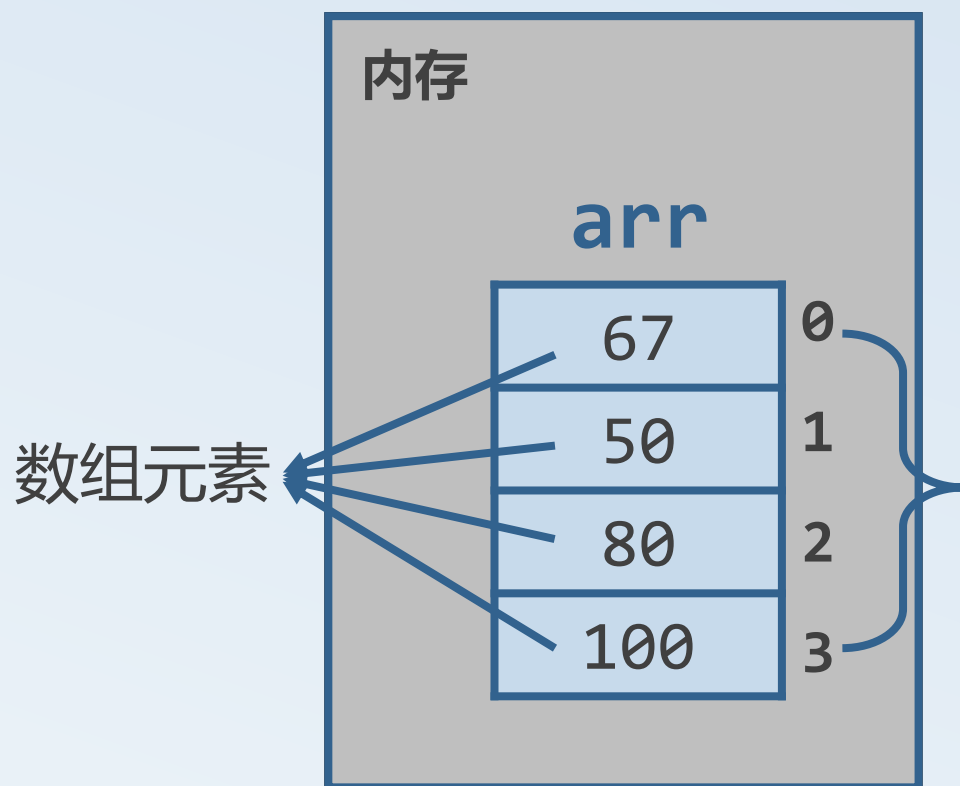
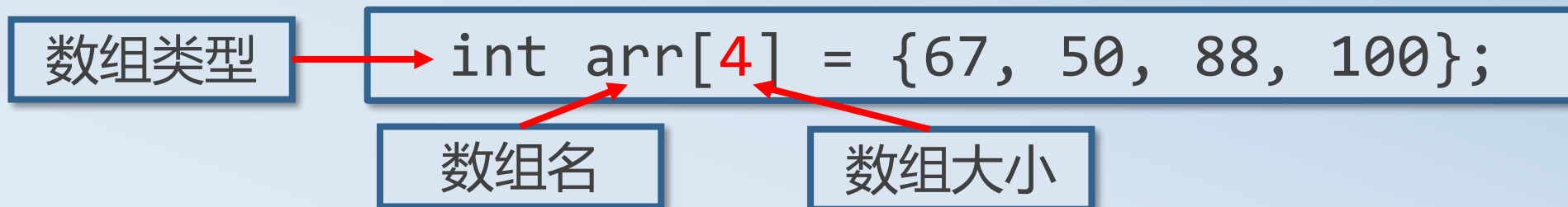
`int arr[10] = {10,9,8,7,6,5,4,3,2,1,0};` ❌

`int arr[10] = {9,8,7,5};` ✅

`int arr[ ] = {9,8,7};` ✅

`int arr[ ] = {};` ❌

# 一维数组



程序中数组的特点：

- ◆ 数组是连续的存储多个元素的结构
- ◆ 数组中所有元素必须具有相同的数据类型

**下标**标明了元素在数组中的位置

# 一维数组

❖例：编写程序从键盘上接收10个整数到一个一维数组中，并完成以下功能：


1. 在屏幕上逐个输出这10个整数
2. 在屏幕上输出最大值、最小值以及平均值
3. 在屏幕上倒序输出10个整数

# 一维数组

## 注意：

- ❖ 数组中必须存放同类型的对象
- ❖ 数组名的命名规则与普通标识符的命名规则相同
- ❖ 数组中对象可以是数值、字符、指针、结构体等类型
- ❖ 数组中的元素还可以是另外一个数组
- ❖ 数组中的元素地址在内存中是连续的

# 本章授课内容



一维数组

二维数组



一维字符数组与字符串



## 二维数组

某个一维数组的每一个元素都是一个一维数组。这种数组我们叫做**二维数组**。

❖格式：

**类型名 数组名[行数][列数]**

❖例如：

`int arr[2][3];` // 定义一个含有2行3列的整型数组

行数

列数

## 二维数组

```
int a[2][3];
```

我们可以将二维数组逻辑的看成一个二维矩阵

第1行	a[0][0]	a[0][1]	a[0][2]	a[0]
第2行	a[1][0]	a[1][1]	a[1][2]	a[1]
	第1列	第2列	第3列	

# 二维数组

## 二维数组的初始化

❖ 按行全部赋初值：

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

❖ 顺序全部赋初值：

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

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

# 二维数组

## 二维数组的初始化

❖ 部分赋初值：（按存储顺序）

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

a[0]	1	2	4
a[1]	0	0	0

# 二维数组

## 二维数组的初始化

❖ 部分赋初值：（按行部分赋值）

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

a[0]	1	2	0
a[1]	4	0	0

- 未赋值的元素均为0
- 要么按行赋值，要么按存储顺序赋值

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



# 二维数组

## 二维数组的初始化

❖ 省略一维长度赋初值

定义时可省略一维长度！

( 按行 ) `int x[ ][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 0, 0 } };`

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

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

( 按存储顺序 ) `int x[ ][3] = { 1, 2, 3, 4, 5, 6, 7 };`

注意：不能省略第二维长度！

## 二维数组

❖例：一个学习小组有5个人，每个人有高数、英语、C语言三门课的考试成绩。求全组分科的平均成绩。

--	高数	英语	C语言
关羽	85	92	87
张飞	61	64	55
赵云	90	92	97
马超	80	87	88
黄忠	89	83	84



# 二维数组

```
#include <stdio.h>
int main(void)
{
    int i,j;
    int arrScore[5][3]; // 存储所有学生成绩
    double arrAve[3];   // 存储科目平均成绩
    int sum;
    printf("请依次输入各科成绩\n");
```

```
    for(i=0; i<5; i++)
    {
        printf("第%d个学生的成绩:", i+1);
        for(j=0; j<3; j++)
        {
            scanf("%d",&arrScore[i][j]);
        }
    }
```

循环接收  
成绩信息

```
    for(i=0; i<3; i++)
    {
        sum = 0;
        for(j=0; j<5; j++)
        {
            sum += arrScore[j][i];
        }
        arrAve[i] = (double)sum / 5;
    }
```

循环计算  
平均信息


```
    printf("数学平均分: %.21f\n",
arrAve[0]);
    printf("英语平均分: %.21f\n",
arrAve[1]);
    printf("C语言平均分: %.21f\n",
arrAve[2]);
    return 0;
}
```

# 二维数组

## ❖注意：

- ◆数组名的命名规则与变量的命名规则相同
- ◆二维数组中的元素的类型必须相同
- ◆二维数组在内存中仍然是按一维数组的方式存储的，先行后列
- ◆ $p[i][j]$ 是数组中第 $i+1$ 行第 $j+1$ 列的元素

# 本章授课内容



一维数组

二维数组

一维字符数组与字符串



# 一维字符数组与字符串

❖ 用于存放字符的一维数组称为一维字符数组。

❖ 字符数组的初始化：

◆ 传统的字符集合方式

```
char ch[5] = {'a', 'b', 'c', 'd', 'e'};
```

◆ 字符串字面值方式

```
char ch[3] = "abcde"; // 不检查越界
```

```
char ch[] = "abc"; <=> char ch[]={'a', 'b', 'c', '\0'};
```

# 一维字符数组与字符串

- ❖ 字符串：连续的字符组成一个串，在内存中存放时，以'\0'为结束标识。字符串的长度就是这个串中字符的个数，但是不包括'\0'。
- ❖ 在C语言中没有专门的字符串变量，通常用一个字符数组来存放一个字符串。
- ❖ 字符数组和字符串的区别是：字符串的末尾有一个空字符 '\0'。

字符串可按如下方式声明并初始化：

```
char name[15] = {'G', 'u', 'a', 'n', 'Y', 'u', '\0'};
```

手工加入一个空字符

```
char name[15] = "GuanYu";
```

系统将自动加入一个空字符

```
char name[] = "GuanYu";
```

省略数组大小，系统自动计算大小为后面的字符总数加1，最后一个元素存入一个空字符。

# 一维字符数组与字符串

- ❖ 字符串字节数：有多少个字符就占多少个字节，包括 '\0'。
- ❖ 字符串的长度：从字符串第一个字符到第一个 '\0' 的位置之前有几个字符。

"GuanYu"

"12345678"

"12\t\n"

"abcd\r\0"

"ab\0ced\r\0"

# 一维字符数组与字符串

```
char name[10];
```

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

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

Guan Yu

Guan

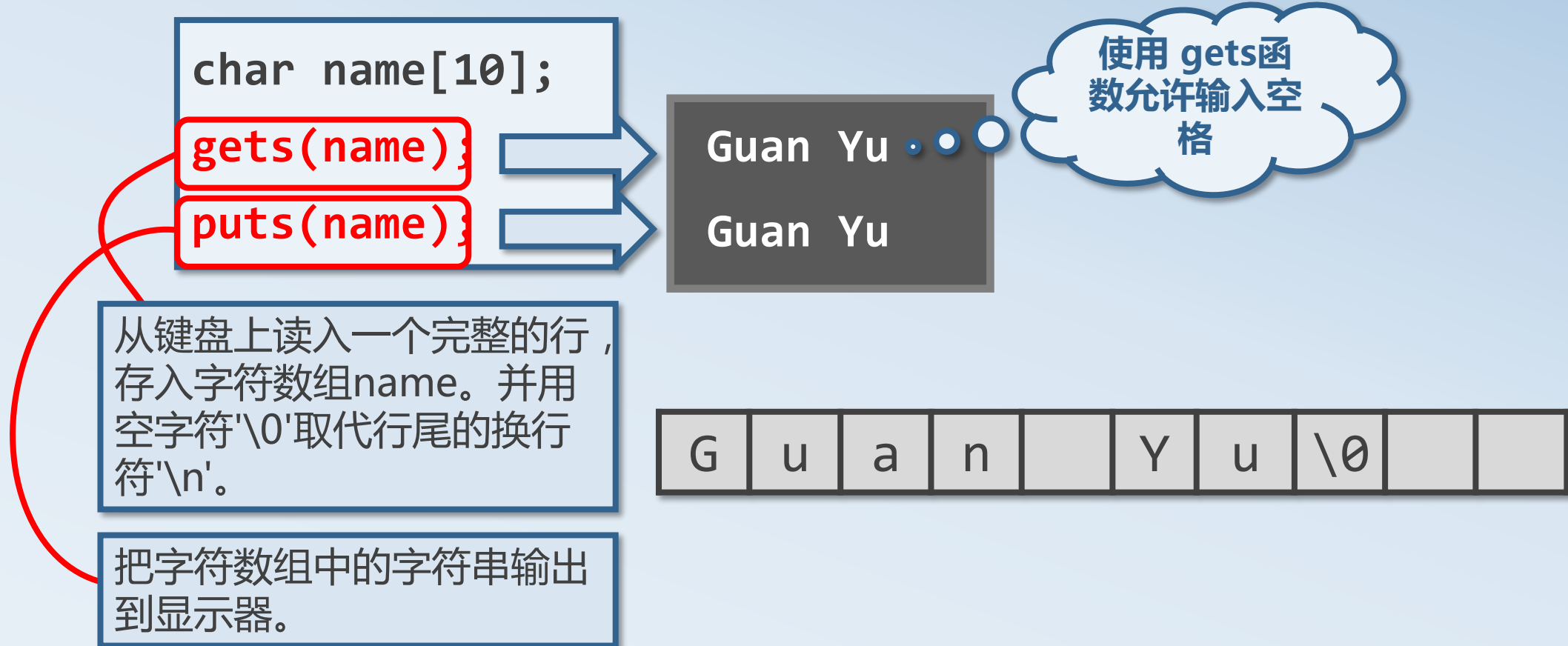
使用 scanf 时，  
不能输入空格

格式描述串中使用转换字符串 "%s"

G	u	a	n	\0					
---	---	---	---	----	--	--	--	--	--



# 一维字符数组与字符串



# 一维字符数组与字符串

- ❖ 字符串处理函数的实现是相同的，所以C标准进行了封装
- ❖ 与字符串有关的内置函数在头文件string.h中定义
- ❖ 要使用标准库字符串处理函数，程序前应该包含：

**#include <string.h>**

**string.h**



- strlen** // 求字符串长度
- strcpy** // 字符串拷贝
- strcmp** // 字符串比较
- strcat** // 字符串拼接
- .....**

# 一维字符数组与字符串

## strlen

语法

```
strlen(s);
```

描述

计算字符串s中字符的个数，并将字符的个数作为函数的返回值。在计算字符个数时不计表示字符串结束的空字符'\0'。

# 一维字符数组与字符串

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

int main(void)
{
    char arr[] = "Beijing";
    int len1, len2;
    len1 = strlen(arr);
    len2 = strlen("Shanghai");
    printf("string = %s length = %d\n", arr, len1);
    printf("string = %s length = %d\n", "Shanghai", len2);

    return 0;
}
```

```
string = Beijing length = 7
string = Shanghai length = 8
```

# 一维字符数组与字符串

## strcpy

语法

```
strcpy(dest, src);
```

描述

其中，dest是目标字符串，src是源字符串。相当于把字符数组src中的字符串拷贝到字符数组dest中。结束标志'\0'也一同拷贝。src可以是一个字符串常量。字符数组dest应足够大，以保证字符串复制不越界。

# 一维字符数组与字符串

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

int main(void)
{
    char source[] = "We change lives";
    char target[20];
    strcpy(target, source);
    printf("源字符串 = %s\n", source);
    printf("目标字符串 = %s\n", target);

    return 0;
}
```

源字符串 = We change lives

目标字符串 = We change lives

# 一维字符数组与字符串

## strcmp

语法

```
strcmp(str1, str2);
```

描述

按照ASCII码顺序比较字符串str1和str2的大小，比较的结果由函数返回。在两个字符串str1和str2相同时返回0；字符串str1大于字符串str2时返回一个正值，否则就返回负值。



# 一维字符数组与字符串

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

int main(void)
{
    char username[15], pwd[15];
    printf("请输入用户名: ");
    gets(username);
    printf("请输入密码: ");
    gets(pwd);
    if((strcmp(username, "John")==0) && (strcmp(pwd, "123456")==0))
        printf("您已成功登录\n");
    else
        printf("用户名或密码无效\n");

    return 0;
}
```

请输入用户名 : john  
请输入密码 : 123456  
用户名或密码无效

请输入用户名 : John  
请输入密码 : 123456  
您已成功登录

# 一维字符数组与字符串

## strcat

语法

```
strcat(dest, src)
```

描述

把字符串 src 中的字符串连接到字符串 dest 中字符串的后面。本函数返回值是字符数组 dest 的首地址。连接后字符串的总长度将是字符串 src 的长度加上字符串 dest 的长度。目标字符串 dest 的大小应足够存储最终的字符串。

# 一维字符数组与字符串

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

int main(void)
{
    char source_string[] = "is very good";
    char target_string[30] = "ACCP 4.0 ";
    strcat(target_string, source_string);
    printf("源字符串 = %s\n", source_string);
    printf("目标字符串 = %s\n", target_string);

    return 0;
}
```

源字符串 = is very good

目标字符串 = ACCP 4.0 is very good

Thank You !