

计算机程序设计基础(1)

09 数组（上）

清华大学电子工程系

杨昉

E-mail: fangyang@tsinghua.edu.cn



- 函数（模块）的递归调用

- 直接递归

- 间接递归

- 递归关键：**递归式+初始条件**

- 编译预处理

- 文件包含命令 [#include](#)

- 条件编译命令 [#if...](#)

- [#pragma](#)

- [#line](#)

课程回顾: 递归与编译预处理



类型	定义	说明
递归	直接递归 : 直接调用函数本身 间接递归 : 通过调用别的函数调用自身	注意递归与操作的 时序问题 。找出递归的 初始值 和 递归表达式
文件包含	<code>#include <文件名></code> (标准方式 检索其他目录) <code>#include "文件名"</code> (所属的文件目录 中寻找所包含的文件, 再按 标准方式 检索其他目录)	将指定文件中的全部内容 插入 到 该命令所在的位置 后一起被编译; 一次只能包含一个, 可嵌套使用, 不可递归包含
条件编译	<code>#ifdef, #else, #endif / #ifdef, #endif</code> <code>#ifndef, #else, #endif / #ifndef, #endif</code> <code>#if, #else, #endif / #if, #endif</code> <code>#if, #elif, #else, #endif / #undef</code>	只在 满足一定条件 时才进行编译; 或者当满足条件时 对一部分语句 进行编译, 可 减少目标程序长度, 减少运行时间 。 <code>#ifdef</code> 只关注 定义本身 , 与 具体值无关
<code>#pragma</code>	<code>#pragma token-string</code>	once 让编译器把指定文件 只包含一次 , warning 让编译器 不再显示这类警告
<code>#line</code>	<code>#line 数字["文件名"]</code>	让编译器编译显示错误信息时, 改变当前所显示的 行号 和 文件名



9.1 数组的定义与引用

- 数组的概念
- 一维数组
- 二维数组
- 数组的初始化

9.2 字符数组与字符串

- 字符数组
- 字符串
- 字符数组与字符串的输入与输出
- 字符串处理函数



9.1 数组的定义与引用

- 数组的概念
- 一维数组
- 二维数组
- 数组的初始化

数组的基本概念



●例9-1：羊称重问题

□假设有1000只羊，如何从羊群中选一只最肥的羊？如何记录每只羊的重量？需要定义1000个变量(a_1, a_2, \dots)吗？

□难点：(1) 如何表示相同类型的序列数据；
(2) 如何实现1000次比较操作？

●数组

□相同数据类型元素的集合

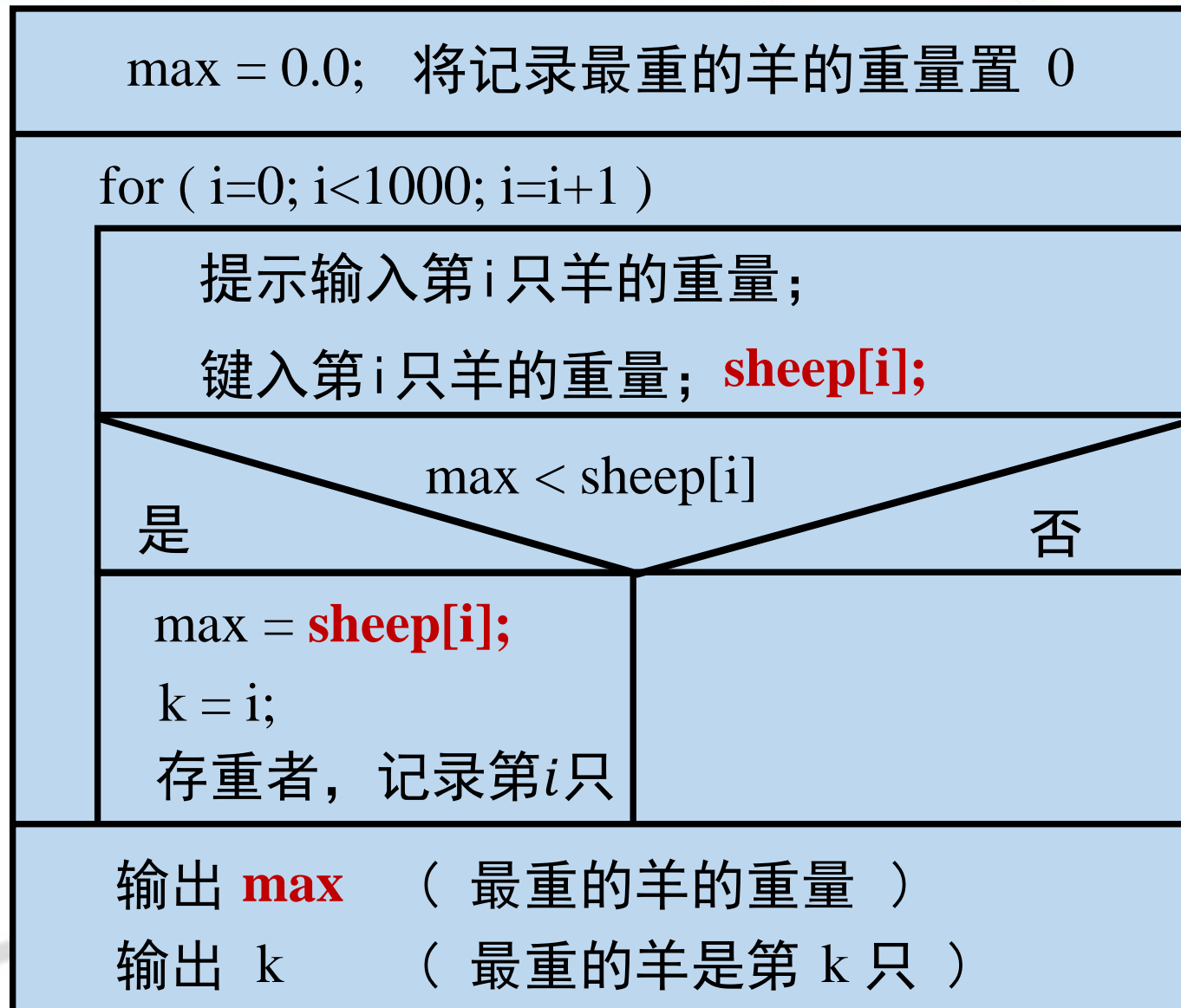
□用统一的数组名表示，数组中的每一个元素通过下标来区分

□数组元素又称为下标变量，而以前所讲的变量称为简单变量

数组的基本概念



●例9-1:羊称重问题



数组的基本概念



●例9-1:羊称重问题

```
1  #include <stdio.h>
2  void main() {
3      float sheep[1000];           //数组定义, 相当于定义1000个变量
4      float max;
5      int i, k;
6      max = 0.0;
7      for (i = 0; i < 1000; i = i + 1) {
8          printf("请输入羊的重量sheep[%d] = ", i);
9          scanf("%f", &(sheep[i])); //数组的使用, 元素单个使用
10         if (max < sheep[i]) {
11             max = sheep[i];       //数组的使用
12             k = i;
13         }
14     }
15     printf("max = %f\n", max);
16     printf("number = %d\n", k);
17 }
```


数组定义与引用：一维数组



●一维数组

□一般形式:

类型说明符 数组名[常量表达式];

□类型说明符：定义数组中各元素的数据类型

□数组名：命名规则与变量名相同

□常量表达式：说明数组的大小（即数组中元素个数）

□C语言中定义数组长度为N，数组元素的下标范围是0到N-1

□C语言中，只能逐个引用数组元素，不能一次引用全部元素

数组定义与引用：一维数组



●一维数组

□一般形式：

类型说明符 数组名[常量表达式];

□常量表达式必须为整型，并且用方括号括起来(不能用圆括号)

□说明数组大小的常量表达式中可以包含符号常量，但不能是变量

```
1 int n;  
2 scanf("%d", &n);  
3 int a[n];
```

错误：因为n是变量，不能用来定义数组

数组定义与引用：一维数组



●例9-2：如何定义数组和引用数组元素

□只能逐个给数组元素赋值

□也只能逐个打印数组元素

□不能整体一起赋值、输入输出

```
1 #include <stdio.h>
2 #define N 5
3 void main() {
4     int i, a[N];
5     for (i = 0; i < N; i++) a[i] = i;
6     for (i = 0; i < N; i++)
7         printf("%5d", a[i]);
8     printf("\n");
9 }
```

0 1 2 3 4
请按任意键继续...

数组定义与引用：二维数组



●二维数组

□一般形式：

类型说明符 数组名[常量表达式1][常量表达式2];

□定义两个二维数组

```
double a[3][4], b[5][10];
```

数组a：双精度实型，3行4列，共有12个元素

数组b：双精度实型，5行10列，共有50个元素

数组定义与引用：二维数组



●二维数组

□一般形式：

类型说明符 数组名[常量表达式1][常量表达式2];

□C语言中，二维数组在计算机中的存储顺序是以行为主的

`double a[3][4];`

□数组a在计算机中逐行存储的顺序如下

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

数组定义与引用：一维数组初始化



●一维数组初始化方法

- 1. 利用输入函数为数组中的各个元素逐个输入值（例9-1）
- 2. 利用赋值语句对数组中的元素进行逐个赋值（例9-2）
- 3. 初始化，即在定义数组时直接为各个元素赋初值（例9-3）

●例9-3：一维数组初始化

```
1 #include <stdio.h>
2 void main() {
3     int a[10] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 }, i;
4     for (i = 0; i < 10; i++)
5         printf("%d, ", a[i]);
6     printf("\n");
7 }
```

1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
请按任意键继续...

数组定义与引用：一维数组初始化



●一维数组初始化说明

- 任何类型的数组（包括auto数组），可以只给前若干个元素赋初值，此时后面的元素均将**自动赋初值0**
- 在对程序进行编译连接时，**外部数组**和**静态数组**就给予分配存储空间，并**自动赋初值**（**0**或者**给定的初值**）
- 程序中定义的动态(auto)数组是在**运行时分配存储空间**，如果没有给任何元素赋初值，其中各元素的初值**随机**

数组定义与引用：一维数组初始化



●例9-4：数组部分赋初值

```
1 #include <stdio.h>
2 void main() {
3     int a[10] = {1, 2, 3, 4, 5}, i; // 后5个元素自动赋初值为0
4     for (i = 0; i < 10; i++)
5         printf("%d, ", a[i]);
6     printf("\n");
7 }
```

1, 2, 3, 4, 5, 0, 0, 0, 0, 0,
请按任意键继续...

数组定义与引用：一维数组初始化



●例9-5：不同类型的数组不赋初值

□x为动态 (auto) 数组，未赋初值时数组中元素为随机数

□y为静态数组，未赋初值时自动赋初值0

□z为动态 (auto) 数组，部分赋初值时，未赋初值的元素为0

```
1 #include <stdio.h>
2 void main() {
3     int k, x[5];
4     static int y[5];
5     int z[5] = { 0 };
6     for (k = 0; k<5; k++)
7         printf("%5d%5d%5d\n", x[k], y[k], z[k]);
8 }
```

```
-858993460  0  0
-858993460  0  0
-858993460  0  0
-858993460  0  0
-858993460  0  0
请按任意键继续...
```

数组定义与引用：一维数组初始化



●一维数组初始化说明

- 在对全部元素赋初值时，说明语句中可以不指定数组长度，其长度默认与初值表中数据的个数相同
- 如果不是对全部元素赋初值，则说明语句中必须说明数组长度
- 早期标准C规定只能对“静态存储”的数组进行初始化，实际上现在可以对任何类型的数组，包括外部(全局)数组、静态(static)数组、和局部(auto)数组进行初始化

数组定义与引用：一维数组初始化



●例9-6：不同类型的数组初始化

```
1 #include <stdio.h>
2 int k, x[5];
3     /* 外部(全局)数组变量会自动初始化为0 */
4 void main() {
5     static int y[5];
6     /* 用static说明的局部静态数组会自动初始化为0 */
7     int z[] = { 0, 0, 0, 0, 0 };
8     /* 定义不写长度, 由初值个数确定数组长度*/
9     for (k = 0; k<5; k++)
10         printf("%5d%5d%5d\n", x[k], y[k], z[k]);
11 }
```

0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

请按任意键继续...

数组定义与引用：一维数组初始化



●例9-7：天数问题

□从键盘输入年、月、日,计算并输出该日是该年的第几天

```
1 #include <stdio.h>
2 void main() {
3     int year, month, day, k, sum;
4     int t[] = { 31, 0, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
5     printf("input year,month,day:");
6     scanf("%d%d%d", &year, &month, &day);
7     if ((year % 4 == 0 && year % 100 != 0) || year%400==0)
8         t[1] = 29;
9     else t[1] = 28;
10    sum = day;
11    for (k = 0; k<month - 1; k++)
12        sum = sum + t[k];
13    printf("Days=%d\n", sum);
14 }
```

input year,month,day:2021 9 5
Days=248
请按任意键继续...

数组定义与引用：二维数组初始化



●二维数组初始化方法

- 1. 利用赋值语句对数组中的元素进行逐个赋值
- 2. 利用输入函数为数组中的各个元素逐个输入值
- 3. 初始化,即在定义数组时直接为各个元素赋初值

●例9-8：二维数组赋初值

```
1 #include <stdio.h>
2 void main() {
3     int a[3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
4     for (int i = 0; i < 3; i++) {
5         for (int j = 0; j < 4; j++)
6             printf("%d ", a[i][j]);
7         printf("\n");
8     }
9 }
```

```
1 2 3 4
5 6 7 8
9 10 11 12
请按任意键继续...
```

数组定义与引用：二维数组初始化



●二维数组初始化说明

- 分行**给二维数组赋初值时，对于每一行都可以只对前几个元素赋初值，后面未赋初值的元素系统将自动赋初值0，并且可以只对前几行元素赋初值
- 在给全部元素赋初值时，说明语句中可以省略第一维的长度说明（但**方括号不能省略**）
- 在分行赋初值时，也可以省略第一维的长度说明

数组定义与引用：二维数组初始化



●例9-9：二维数组分行赋初值

```
1 #include <stdio.h>
2 void print_num(int a[3][4]) {
3     for (int i = 0; i < 3; i++) {
4         for (int j = 0; j < 4; j++)
5             printf("%d ", a[i][j]);
6         printf("\n");
7     }
8 }
9 void main() {
10     int a[3][4] = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}; //分行赋初值
11     int b[3][4] = {{1, 2}, {5}, {9, 10, 11}}; //每行中前几个元素赋初值
12     print_num(a);
13     printf("\n");
14     print_num(b);
15 }
```

```
1 2 3 4
5 6 7 8
9 10 11 12
```

```
1 2 0 0
5 0 0 0
9 10 11 0
请按任意键继续...
```


数组定义与引用：二维数组初始化



●例9-10：二维数组赋初值省略第一维长度

```
1 #include <stdio.h>
2 void print_num(int a[3][4]) {
3     for (int i = 0; i < 3; i++) {
4         for (int j = 0; j < 4; j++)
5             printf("%d ", a[i][j]);
6         printf("\n");
7     }
8 }
9 void main() {
10     int c[][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13 };
11     //等价于int a[4][4], 最后3个元素值为0
12     int d[][4]={ {1, 2}, {5}, {9, 10, 11} }; //等价于int a[3][4], 缺省者为0
13     print_num(c);
14     printf("\n");
15     print_num(d);
16 }
```

```
1 2 3 4
5 6 7 8
9 10 11 12
13 0 0 0
```

```
1 2 0 0
5 0 0 0
9 10 11 0
```

请按任意键继续...

课堂练习



练习1：写出右侧代码运行结果

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix} \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

70 80 90
158 184 210
请按任意键继续...

```
1 #include <stdio.h>
2 void main() {
3     int i, j, k, c[2][3];
4     int a[2][4] = { 1, 2, 3, 4, 5, 6, 7, 8 };
5     int b[4][3] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
6     for (i = 0; i<2; i++) /*双重循环*/
7         for (j = 0; j<3; j++) {
8             c[i][j] = 0; /*赋初值*/
9             for (k = 0; k<4; k++)
10                 c[i][j] = c[i][j] + a[i][k] * b[k][j];
11         }
12     for (i = 0; i<2; i++) { /*输出*/
13         for (j = 0; j<3; j++)
14             printf("%6d", c[i][j]);
15         printf("\n");
16     }
17 }
```

Debug tips

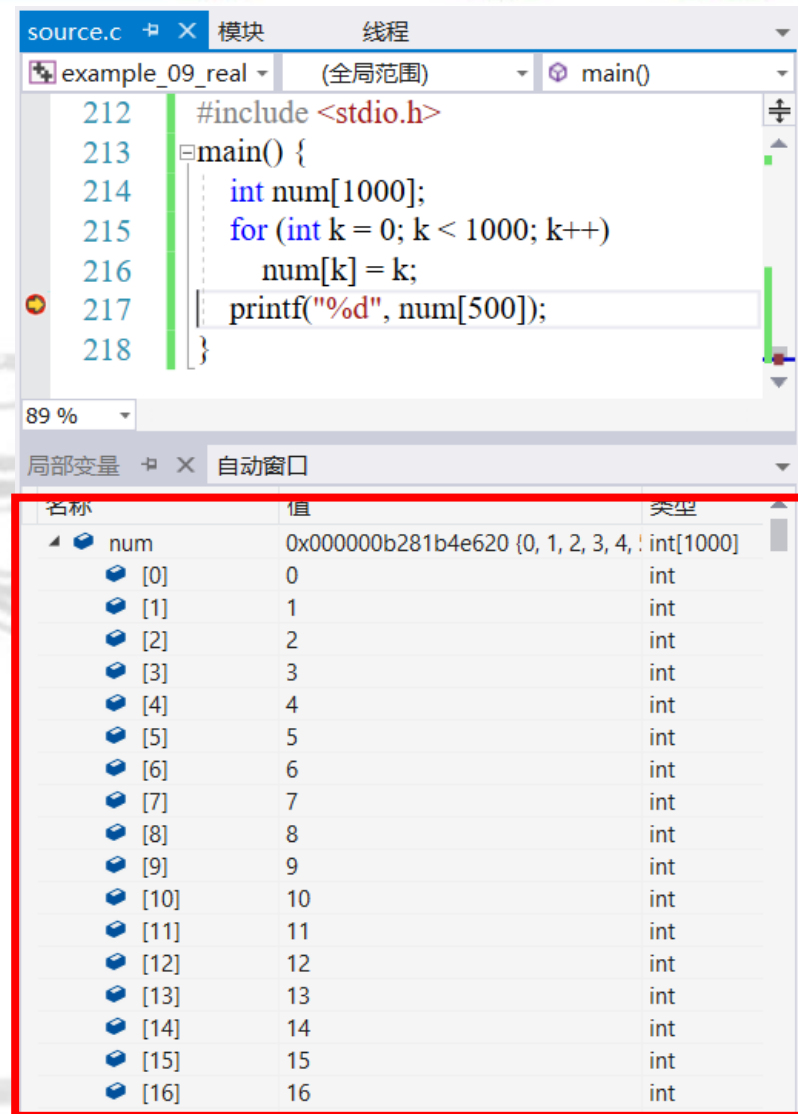


● Debug过程中查看数组元素

□ 老办法：断点+查看局部变量

□ 如果数组较大，查看数组中特定位置上的元素仍然比较麻烦，如何解决？

```
1 #include <stdio.h>
2 void main() {
3     int num[1000];
4     for (int k = 0; k < 1000; k++)
5         num[k] = k;
6     printf("%d", num[500]);
7 }
```



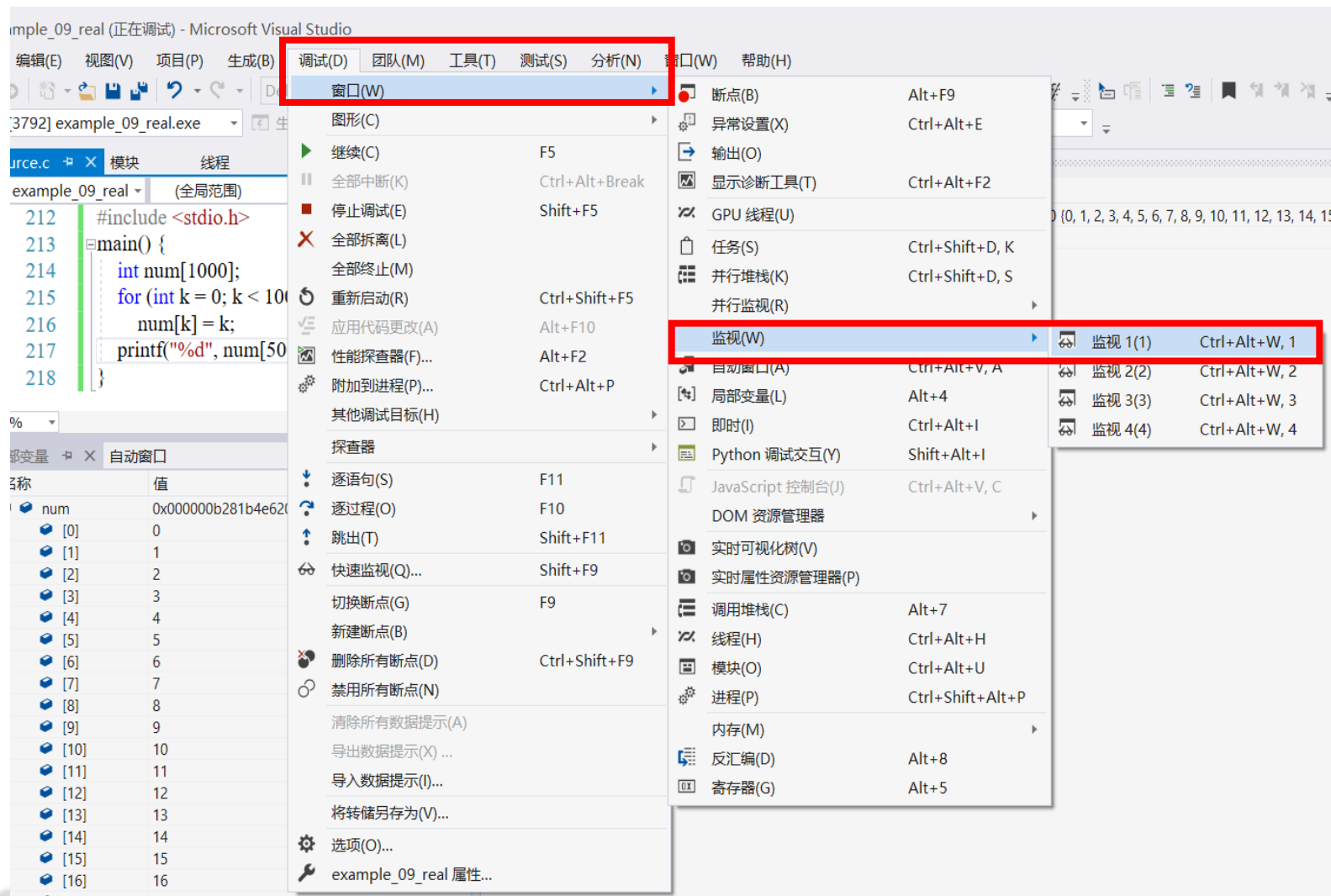
Debug tips



● Debug过程中查看数组元素

□ 添加**监视**

□ 调试→窗口→监视
→监视1（或者监视2/3/4）



Debug tips

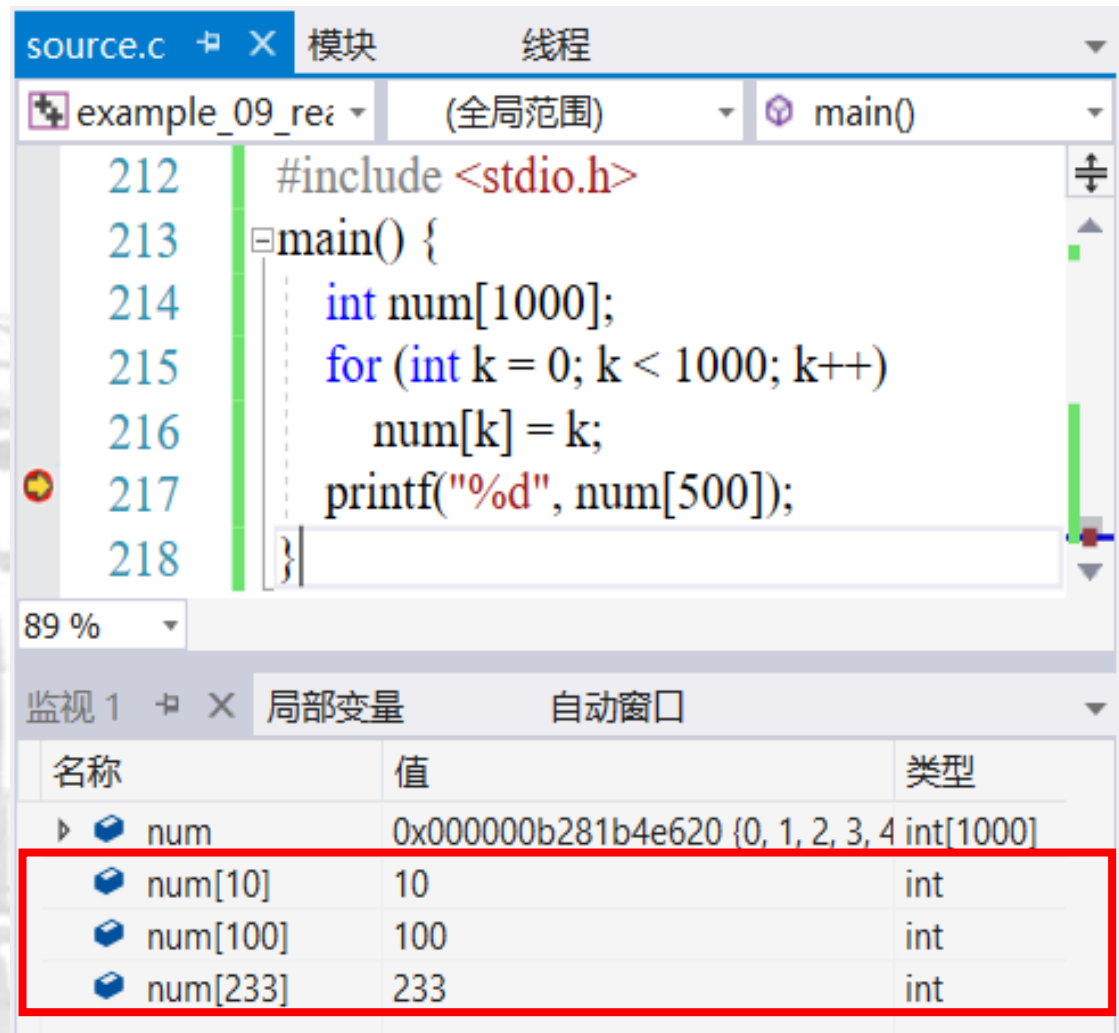


● Debug过程中查看数组元素

□ 添加**监视**

□ 调试→窗口→监视→监视1（或者监视2/3/4）




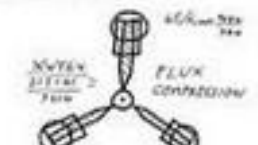

□ 在监视窗口中输入**想要查看的数组元素**



Debug tips



●Teach Yourself C++ in 21 Days

<p>第1-10天</p> <p>学习变量，常量，数组，字符串，表达式，语句，函数……</p> 	<p>第11-21天</p> <p>学习程序流程，指针，引用，类，对象，接口，多态……</p> 	<p>第22-697天（两年）</p> <p>进行大量的“休闲娱乐”方式的编程，并在Hack代码中找到乐趣，从错误中学习总结。</p> 
<p>第698-3648天（八年）</p> <p>与其它程序员互相影响，并一同开发项目，并向他们学习。</p> 	<p>第3649-7781天（十年）</p> <p>学习高等理论物理，使用公式证明量子物理理论。</p> 	<p>第7782-14611天（二十年）</p> <p>学习生物化学，分子生物学，以及遗传学。</p> 
<p>第14611天 （自学习编程来的四十年）</p> <p>使用在生物学方面的知识造一个“返老还童”药剂。</p> 	<p>第14611天</p> <p>使用在物理学上的知识造一个通量电容器，把自己传送回学编程的第21天。</p> 	<p>第21天</p> <p>把那时的自己给替换了。</p> 



9.2 字符数组与字符串

- 字符数组
- 字符串
- 字符数组与字符串的输入与输出
- 字符串处理函数

字符数组与字符串：字符数组



●字符数组

□用于存放字符型数据的数组

□在C语言中,字符数组中的一个元素只能存放一个字符

●字符数组定义

□一维字符数组:

`char 数组名[常量表达式];`

□二维字符数组:

`char 数组名[常量表达式1][常量表达式2];`



●字符数组初始化

- 当对字符数组中所有元素赋初值时,数组的长度说明可以省略
- 可以只对前若干元素赋初值
- 早期标准C规定只能对静态字符数组初始化,但目前在大多数的编译系统中,也可以对外部和局部(auto)字符数组进行初始化

字符数组与字符串：字符数组



●例9-11：字符数组赋初值

```
1 #include<stdio.h>
2 void main() {
3     char a[6] = { 'h', 'e', 'l', 'l', 'o', '\0' };
4     char b[10] = { 'h', 'e', 'l', 'l', 'o', '\0' }; //对前6个元素赋初值
5     char c[] = { 'h', 'e', 'l', 'l', 'o', '\0' }; //省略数组长度说明
6     for (int k = 0; k < 6; k++)
7         printf("%c", a[k]);
8     printf("\n");
9     for (int k = 0; k < 10; k++)
10        printf("%c", b[k]);
11    printf("\n");
12    for (int k = 0; k < 6; k++)
13        printf("%c", c[k]);
14    printf("\n");
15 }
```

```
hello
hello
hello
请按任意键继续...
```



●字符串说明

- 字符串常量（简称字符串）要用一对双撇号括起来
- 在一个字符串常量中，最后还隐含包括一个结束符'\0'
- 例如，“how do you do?”表面上是一个长度为14的字符串常量，但实际包含15个字符，最后一个为结束符'\0'

字符数组与字符串：字符串



●字符串

□C语言允许用字符串常量对字符数组进行初始化

□下列语句等价

1	<code>char a[15] = { "how do you do?" };</code>
2	<code>char a[15] = "how do you do?";</code>
3	<code>char a[] = "how do you do?";</code>
4	<code>char a[] = { 'h', 'o', 'w', ' ', 'd', 'o', ' ', 'y', 'o', 'u', ' ', 'd', 'o', '?', '\0' };</code>
5	<code>char a[] = { 104, 111, 119, 32, 100, 111, 32, 121, 111, 117, 32, 100, 111, 63, 0 };</code>

□下列语句不等价

1	<code>char b[15] = "China";</code> //数组b长度为15
2	<code>char b[] = "China";</code> //数组b长度为6

字符数组与字符串：字符串



●字符串

□利用字符串常量可以对字符数组进行初始化

□不能用字符串常量为字符数组赋值

□例：下列语句正确

```
char b[15]="China";
```

下列用法错误

```
char b[15]; b= "China";
```

```
char b[15]; b[15]= "China";
```



●字符数组的输入与输出

- 对数组中的每一个字符元素逐个进行输入或输出
- 将数组中的所有字符作为一个字符串进行输入或输出
- 格式说明符
 - 格式符%c用于输入输出一个字符
 - 格式符%s用于输入输出一个字符串



●字符数组的输入与输出

□输入输出一个字符（格式说明符为%c）

□在用%c进行输入时，输入项为数组元素地址

□在具体输入时，各字符之间不需要分隔符分隔，字符也不要
用单撇号括起来

□在用%c进行输出时，输出项为数组元素



●例9-12：字符数组的输入与输出

□在下列C程序中,首先分别为字符数组元素a[1]与a[2]读入字符,然后输出数组元素a[2]中的字符

```
1 #include <stdio.h>
2 void main() {
3     char a[5];
4     scanf("%c%c", &a[1], &a[2]);
5     a[0] = 'a'; a[3] = 'd'; a[4] = '\0';
6     printf("%c\n", a[2]);
7 }
```

```
bcgh
c
请按任意键继续...
```



●字符数组的输入与输出

- 输入输出一个字符串（格式说明符为%s）

- 在用%s进行输入输出时,其输入输出项均为数组名

- 在输入时，相邻两个字符串之间要用空格(回车)分隔,系统将自动地在所读入的字符串最后加结束符'\0'

- 在输出时，遇结束符'\0'作为输出结束标志

字符数组与字符串：输入输出



●例9-13：字符数组的输入与输出

□输入输出语句使用%s

```
1 #include <stdio.h>
2 void main() {
3     char a[5];
4     scanf("%c%c", &a[1], &a[2]);
5     a[0] = 'a'; a[3] = 'd'; a[4] = '\0';
6     printf("%s\n", a);
7     char b[20], c[20];
8     scanf("%s%s", b, c);
9     printf("%s\n", b);
10    printf("%s\n", c);
11 }
```

```
bc
abcd
helloworld
debug
helloworld
debug
请按任意键继续...
```



●字符数组的输入与输出

□为字符型数组输入字符串时，输入字符串的长度不能大于数组长度

□如果输入字符串长度大于数组长度，程序运行中会发生错误！

□提醒：字符串中还有一个字符串结束符'\0'，它虽然不显式地出现在字符串中，但它实际需要占一个字节空间，计算时不可忽略

字符数组与字符串：输入输出



●例9-14：字符数组的输入与输出

□对数组进行输入与输出操作

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

```
ab cd
a=ab,b=cd
请按任意键继续...
```

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	b[0]	b[1]	b[2]	b[3]	b[4]	b[5]			
a	b	\0				c	d	\0						

字符数组与字符串：输入输出



●例9-15：字符数组的输入与输出

□对数组进行输入与输出操作

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

abcd efghkmn
a=abcd,b=efghkmn
请按任意键继续...

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	b[0]	b[1]	b[2]	b[3]	b[4]	b[5]			
a	b	c	d	\0		e	f	g	h	k	m	n	\0	

Microsoft Visual C++ Runtime Library



Debug Error!

Program: ...3_助教\计算机程序设计基础
\code\example_09_real\Debug\example_09_real.exe
Module: ...3_助教\计算机程序设计基础
\code\example_09_real\Debug\example_09_real.exe
File:

Run-Time Check Failure #2 - Stack around the variable 'b' was corrupted.

(Press Retry to debug the application)

中止(A)

重试(R)

忽略(I)

程序会报错：输入数组b的内容超出数组b的大小

字符数组与字符串：输入输出



●例9-16：字符数组的输入与输出

□对数组进行输入与输出操作

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

```
abcdefgh kmnp
a=abcdefgh,b=kmnp
请按任意键继续...
```

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]			b[0]	b[1]	b[2]	b[3]	b[4]	b[5]
a	b	c	d	e	f	g	h	k	m	n	p	\0	

Microsoft Visual C++ Runtime Library



Debug Error!

Program: ...3_助教\计算机程序设计基础
\code\example_09_real\Debug\example_09_real.exe
Module: ...3_助教\计算机程序设计基础
\code\example_09_real\Debug\example_09_real.exe
File:

Run-Time Check Failure #2 - Stack around the variable 'a' was corrupted.

(Press Retry to debug the application)

中止(A)

重试(R)

忽略(I)

程序会报错：输入数组a的内容超出数组a的大小



●字符数组的输入与输出

□在用格式说明符`%s`为字符型数组输入数据时，字符串的分隔符是空格符

□如果在输入的字符串中包括空格符时，只截取空格前的部分作为字符串赋给字符数组

□例如，如果从键盘为字符数组输入

Hello world

最终数组内容为 Hello

字符数组与字符串：输入输出



●例9-17：对数组进行输入，输入语句中含有空格

```
1 #include <stdio.h>
2 void main() {
3     char str1[] = "how do you do";
4     char str2[20];
5     scanf("%s", str2);
6     printf("%s\n", str2);
7     printf("%s\n", str1);
8 }
```

```
HOW DO YOU DO
HOW
how do you do
请按任意键继续...
```

字符串处理函数: puts



● puts(字符数组名)

□ 输出一个字符串到终端

□ 引用头文件<string.h>

● 例9-18: puts()使用

```
1 #include <string.h>
2 void main() {
3     char str[] = "China\nBeijing";
4     puts(str);
5 }
```

```
China
Beijing
请按任意键继续...
```

字符串处理函数: gets



● gets(字符数组名)

- 从终端读入一行字符到字符数组，并返回字符数组的地址
- 字符包括空格、制表符，直到遇到回车符
- 引用头文件<string.h>

● 例9-19: gets()使用

```
1 #include <string.h>
2 void main() {
3     char s[80];
4     gets(s);
5     puts(s);
6 }
```

```
ab cdef
ab cdef
请按任意键继续...
```

字符串处理函数: strcat



●strcat(字符数组1,字符串2)

- 将字符串2连接到字符串1的后面，并返回字符串1的地址
- 字符数组1的长度必须足够大，以便能容纳被连接的字符串2
- 连接后系统将自动取消字符串1后面的结束符'\0'
- “字符串2”可以是字符数组名或字符串常量,如

```
strcat(s1,s2);
```

```
strcat(s1,"cdef");
```

- 可以直接printf("%s\n",strcat(s1,s2));
- 引用头文件<string.h>

字符串处理函数: strcat



●例9-20: strcat()使用

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char s1[20] = "abcd";
5     char s2[] = "cdef";
6     strcat(s1, s2);
7     printf("%s\n", s1);
8 }
```

abcdcdef
请按任意键继续...

字符串处理函数: strcpy



●strcpy(字符数组1,字符串2)

- 将字符串2拷贝到字符数组1中
- 字符数组1的长度必须足够大，以便能容纳字符串2
- “字符串2”可以是字符数组名或字符串常量,如

```
strcpy(s1,s2);
```

```
strcpy(s1,"cdef");
```

- 字符串只能用拷贝函数strcpy赋值，不能用赋值语句赋值
- 单个字符可以用赋值语句赋给字符变量或字符数组元素
- 引用头文件<string.h>

字符串处理函数: strcpy



●例9-21: strcpy()使用

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char s1[10] = "abcde";
5     char s2[] = "efgh";
6     strcpy(s1, s2); // 不能s1=s2
7     printf("%s\n", s1);
8 }
```

efgh
请按任意键继续...

字符串处理函数: strncpy



●strncpy(字符数组1,字符串2,n)

□将字符串2前n个字符复制到字符数组1中

□strncpy函数只复制指定的前n个字符，不复制也不自动添加字符串结束符'\0'，需要自己去添加字符串结束符'\0'

●例9-22: strncpy()使用

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char s2[] = "abcde"; char s1[10];
5     strncpy(s1, s2, 3);
6     s1[3] = '\0';
7     printf("%s\n", s1);
8 }
```

abc
请按任意键继续...

字符串处理函数: strcmp



● strcmp(字符串1,字符串2)

□按照字典序比较两个字符串大小

□返回值

情况	返回值
字符串1 == 字符串2	0
字符串1 > 字符串2	正整数
字符串1 < 字符串2	负整数

□“字符串1”与“字符串2”可以是字符数组名或字符串常量

□对字符串不能直接用关系运算符==、!=、>、>=、<、<=进行比较，若a和b是字符串，if (a>b) 是错误的

□引用头文件<string.h>

字符串处理函数: strcmp



● strcmp(字符串1,字符串2)

□ 执行函数时,自左到右逐个比较两个字符串对应位置字符的ASCII码值,直到发现了不同字符或字符串结束符'\0'为止

□ 以最后一个不相同字符的ASCII码值的大小决定两个字符串的大小

● 例9-23: strcmp()使用

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char a[6] = "abc", b[6] = "abcd";
5     if (strcmp(a, b)>0) printf("%s > %s\n", a, b);
6     else printf("%s <= %s\n", a, b);
7 }
```

abc <= abcd
请按任意键继续...

字符串处理函数: strlen



●strlen(字符串1)

- 求字符串长度(不包括结束符'\0')
- “字符串”可以是字符数组名,也可以是字符串常量
- 引用头文件<string.h>

●例9-24: strlen()使用

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

```
5
10, 5
请按任意键继续...
```

字符串处理函数: strlwr,strupr



●strlwr(字符串)

□将字符串中大写字母转换成小写字母

●strupr(字符串)

□将字符串中小写字母转换成大写字母

●例9-25: strlwr()与strupr()的使用

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char a[10] = "abcde";
5     char b[10] = "ABCDE";
6     printf("%s\n", strupr(a));
7     printf("%s\n", strlwr(b));
8 }
```

ABCDE
abcde
请按任意键继续...

字符串处理函数: sprintf



● sprintf(字符数组名, "输出格式", 变量列表)

□ 将结果输出到字符数组中

□ 类似于printf输出到屏幕上

● 例9-26: sprintf()使用

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char str[50];
5     int k = 20;
6     double f = 123.4;
7     sprintf(str, "k=%4d f=%8.3f", k, f);
8     puts(str);
9 }
```

k= 20 f= 123.400
请按任意键继续...

字符串处理函数: sscanf



●sscanf(字符数组名, "输入格式", 变量列表)

□从字符串数组中读入数据

□对应scanf从键盘上读入

●例9-27: sscanf()使用

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char str[50];
5     int k = 20, m;
6     double f = 123.4, d;
7     sprintf(str, "k=%4d f=%8.3f", k, f);
8     sscanf(str, "k=%d f=%lf", &m, &d);
9     printf("m=%4d d=%8.3f\n", m, d);
10 }
```

m= 20 d= 123.400
请按任意键继续...

字符串处理函数



●例9-28：把字符串中的每个空格替换为"%20"

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char str[50], str1[50];    gets(str);
5     int k = 0, len = strlen(str) + 1; //strlen不含'\0'
6     for (int i = 0; i < len; i++) { //对str中的字符进行遍历
7         if (str[i] == ' ') { //遇到空格，在str1中插入%20
8             str1[k] = '%'; str1[k + 1] = '2'; str1[k + 2] = '0';
9             k = k + 3;
10        } else { //str[i]非空格，将str[i]插入str1
11            str1[k] = str[i]; k++;
12        }
13    }
14    puts(str1);
15 }
```

We are happy
We%20are%20happy
请按任意键继续...

练习2: 请写
出输入为
aaaabbccdddee
时, 代码的输
出结果

字符串压缩

aaaabbccdddee

a4b2c3d3e2

请按任意键继续...

```
1 #include <stdio.h>
2 #include <string.h>
3 void main() {
4     char str[50], str1[50];
5     gets(str);
6     int count = 0, k = 0;
7     char last = str[0];
8     for (int i = 0; i < strlen(str)+1; i++) {
9         if (str[i] == last) {
10             count++; continue;
11         }
12         str1[k] = last; str1[k+1] = '0' + count; k = k + 2;
13         last = str[i]; count = 1;
14     }
15     str1[k] = '\0'; puts(str1);
16 }
```



●例9-29：电子邮件转发地址判断问题

- ❑ 每个有效的电子邮件地址都由一个本地名和一个域名组成，以'@'符号分隔。本地名可以含有一个或多个'.'或'+'。例如，在alice@tsinghua.edu.cn中，alice是本地名，tsinghua.edu.cn是域名
- ❑ 如果**在本地名中添加'.'**，则邮件将会转发到本地名中没有'.'的同一地址：例如，"alice.s@tsinghua.edu.cn" 和 "alices@tsinghua.edu.cn" 会转发到同一电子邮件地址
- ❑ 如果**在本地名中添加'+'**，则会忽略第一个加号后面的所有内容：例如 "alice+s@tsinghua.edu.cn" 将转发到alice@tsinghua.edu.cn
- ❑ 两条规则可以同时应用于本地名，但都不适用于域名

●例9-29：电子邮件转发地址判断问题

□ 需要计算s1和s2各自被转发到的地址，为此可以编写一个函数

```
1 int addr_trans(char source[], char target[], int max_len) {
2     int i, j = 0, flag = 0;
3     for(i = 0; i < max_len && source[i] != '@'; i++) {
4         if (source[i] == '.')
5             continue;
6         else if (source[i] == '+') {
7             flag = 1;
8             continue;
9         } else if (flag == 0) {
10            target[j] = source[i];
11            j++;
12        }
13    } //处理@之前的部分
14    for(; i < max_len && source[i] != '\0'; i++) {
15        target[j] = source[i];
16        j++;
17    } //处理@之后的部分
18    return j;
19 }
```

数组作为函数参数
(地址结合)下节课讲

字符串处理函数



●例9-29：电子邮件转发地址判断问题

□ 用主函数调用上述转化函数，实现判断

```
1 void main() {  
2     char s1[50] = "", s2[50] = "", t1[50] = "", t2[50] = "";  
3     int max_len_s1, max_len_s2, max_len_t1, max_len_t2;  
4     scanf("%s", s1);  
5     max_len_s1 = strlen(s1);  
6     scanf("%s", s2);  
7     max_len_s2 = strlen(s2);  
8     max_len_t1 = addr_trans(s1, t1, max_len_s1);  
9     max_len_t2 = addr_trans(s2, t2, max_len_s2);  
10    if(strcmp(t1, t2) == 0) printf("s1 == s2\n");  
11    else printf("s1 != s2\n");  
12 }
```

```
bob.s@tsinghua.edu.cn  
bobs+alice@tsinghua.edu.cn  
s1 == s2  
请按任意键继续...
```

```
bob.s@tsinghua.edu.cn  
bob+alice@tsinghua.edu.cn  
s1 != s2  
请按任意键继续...
```

本节总结



- 数组的基本概念

- 数组的定义与引用

- 一维数组 `int a[10];` 二维数组 `int a[10][10];`

- 数组初始化: 赋值语句、输入函数、初始化

- 字符数组与字符串

- 字符数组的定义与初始化 `char a[10];`

- 字符串定义, 字符数组与字符串的输入与输出

- 字符串处理函数: puts, gets, strcat, strcpy, strncpy, strcmp, strlen, strlwr,strupr, sprintf, sscanf



●作业9

□课本第九章习题2,5,7,10

□完成后将word文档或拍照提交到网络学堂



附加作业



●字符串函数实现

□请编写C程序，实现课件中提到的strcat, strcpy, strncpy, strcmp, strlen,strupr, strlwr等功能

●子字符串查找

□请用C语言编写程序，对于输入的长字符串a和短字符串b，输出b在a中出现的次数。注意：考虑b移动的情况，例如字符串“aa”在字符串“baaaab”中出现了3次

●后缀子字符串排列

□后缀子字符串：例如字符串apple，其后缀子字符串分别为e, le, ple, pple, apple

□请编写C程序，对于任意给定的字符串a，对a的所有后缀子字符串进行字典序排列，并按照从小到大的顺序输出各子串

THANKS