第9讲 数组

深圳大学计算机系

问题：有如下几组数据，它们分别该如何存储呢？

一个班学生的学习成绩
一行文字
一个矩阵

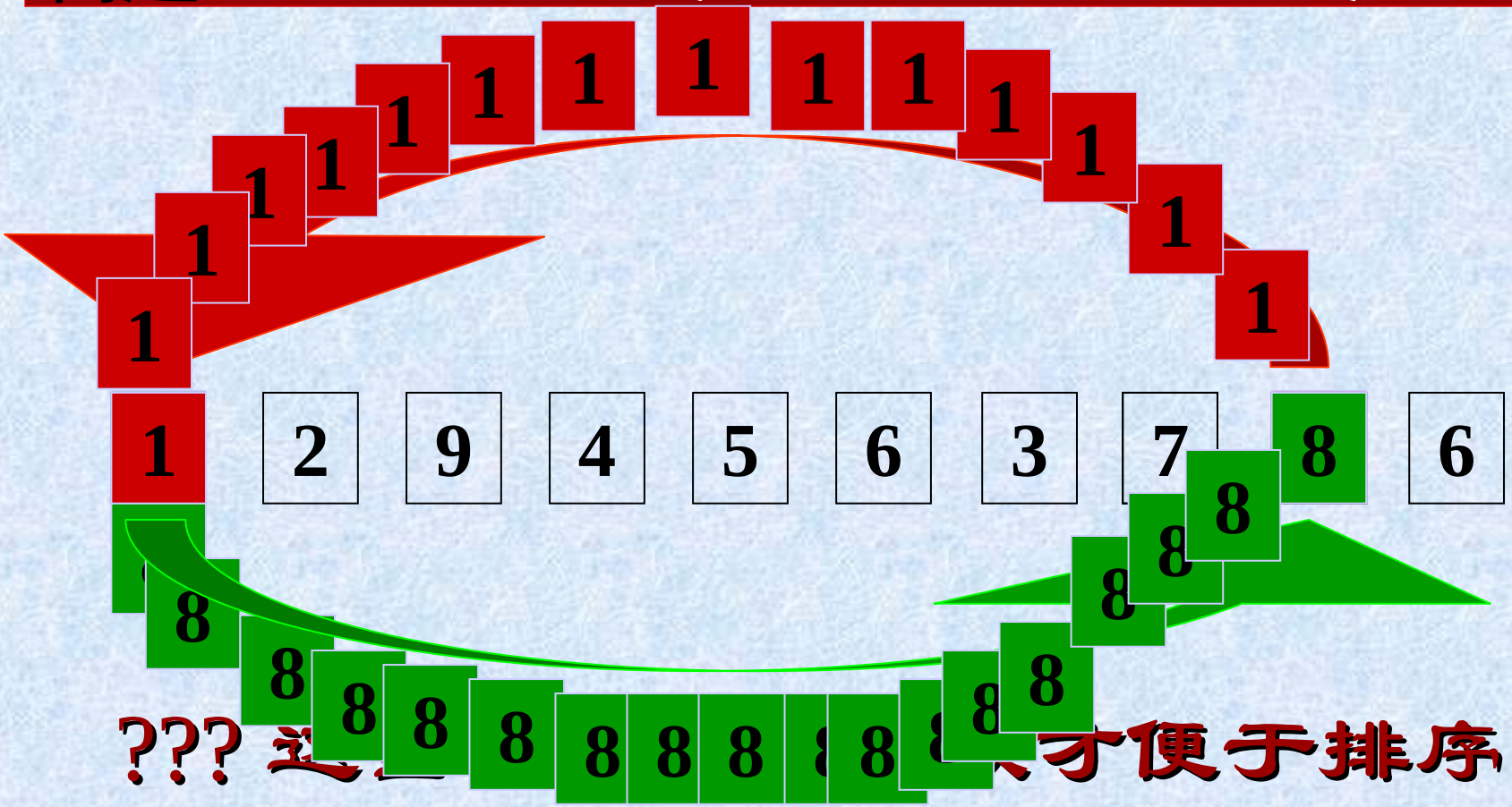
这些数据的特点是：
1、具有相同的数据类型
2、使用过程中需要保留原始数据

C语言为这些数据，提供了一种构造数据类型：数组。

数组

是一组具有相同数据类型的数据的有序集合。

问题：给一组数排序，这组数该如何存放呢



??? 这样才便于排序

这便是本章所要解决的问题

思考

有没有更好的方法来解决呢？

太复杂了！晕！！！

1.问题：

假设我们将100个数分别存放在100个变量中，要计算100个变量的和，如何做？ 能否使用循环语句？

for(sum=0, i=0; i<100; i++) sum=sum+ai; 正确吗？

2. *使用数组解决问题*

- 数组是同类型变量的集合，共用一个名字，用下标区分；
- 每个变量称作数组元素；
- 按下标递增顺序在内存中存放；
- 一维数组与数学中的数列对应，二维数组与矩阵对应。

一维数组

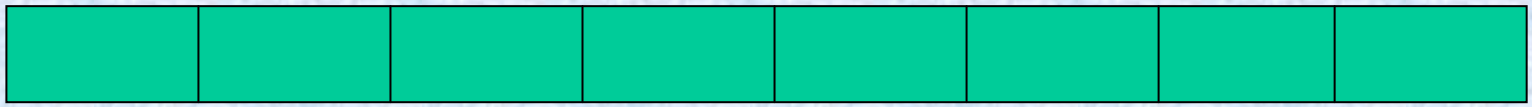
- 数组声明包括：元素类型，数组名，数组元素个数

Element-type Array-Name[size]

- 例:

```
int a[8];
```

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



下标值的范围：**0 ~ 数组大小-1**

➢ 定义说明：

（1）数组定义时，必须指定数组的大小（或长度），数组大小必须是整型常量表达式，不能是变量或变量表达式。

（2）数组定义后，系统将给其分配一定大小的内存单元，其所占内存单元的大小与数组元素的类型和数组的长度有关。

数组所占内存单元的字节数 = 数组大小 × sizeof（数组元素类型）

（3）数组中每个数组元素的类型均相同，它们占用内存中连续的存储单元，其中第一个数组元素的地址是整个数组所占内存块的低地址，也是数组所占内存块的首地址，最后一个数组元素的地址是整个数组所占内存块的高地址（末地址）。

例如： **short int a[20];**

则数组a所占内存单元的大小为：

20 * sizeof(short) = 20 * 2 = 40（字节）。

2、一维数组的引用

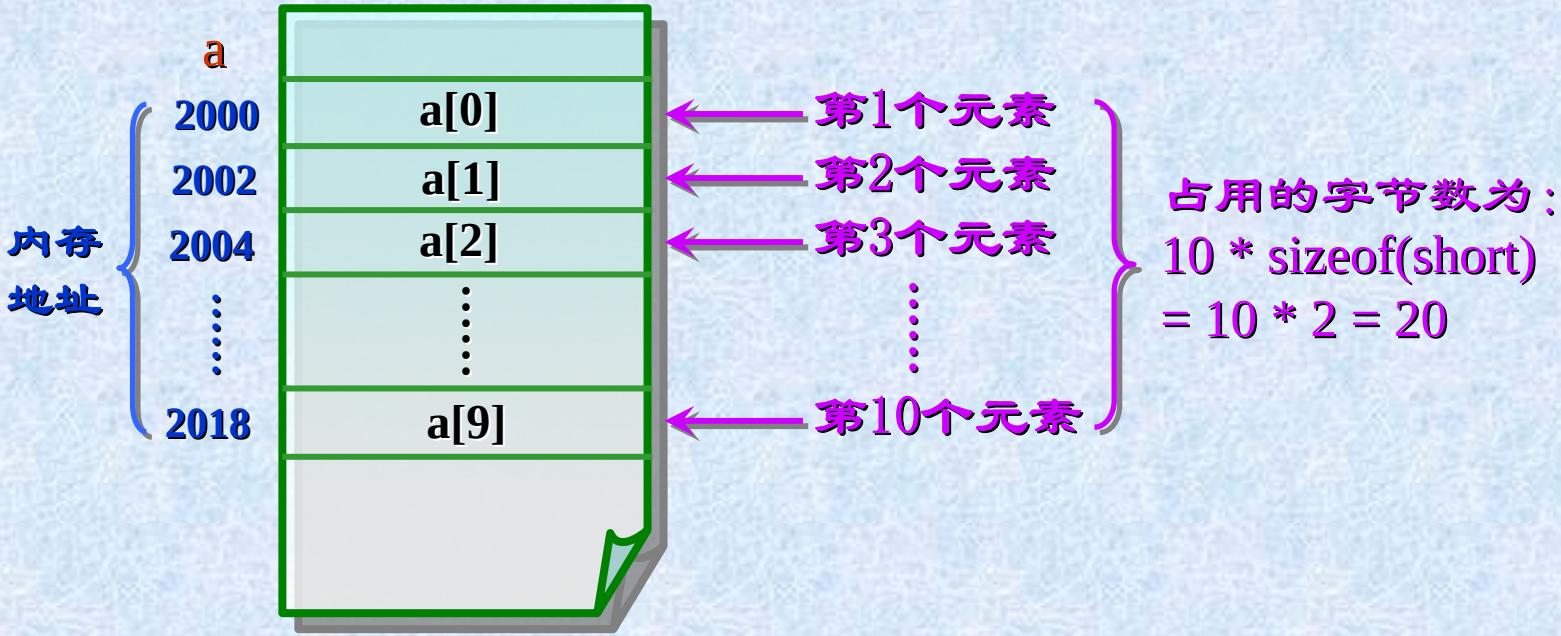
- 引用格式：

数组变量名[下标]

- 引用说明：

（1）下标可以是整型常量、整型变量或整型表达式。C语言规定，下标的**最小值是0，最大值则是数组大小减1**。

例：short int a[10];



- 练习：判断正误

float num[10] ;

(1) num[0] =10;

(2) num[10] = 10.7;

(3) num = 5

(4) num[2] = 3; num[1] = 5;

num[9] = num[2] + num[1];

- 练习：判断正误

1) int N = 10;

float A[N];

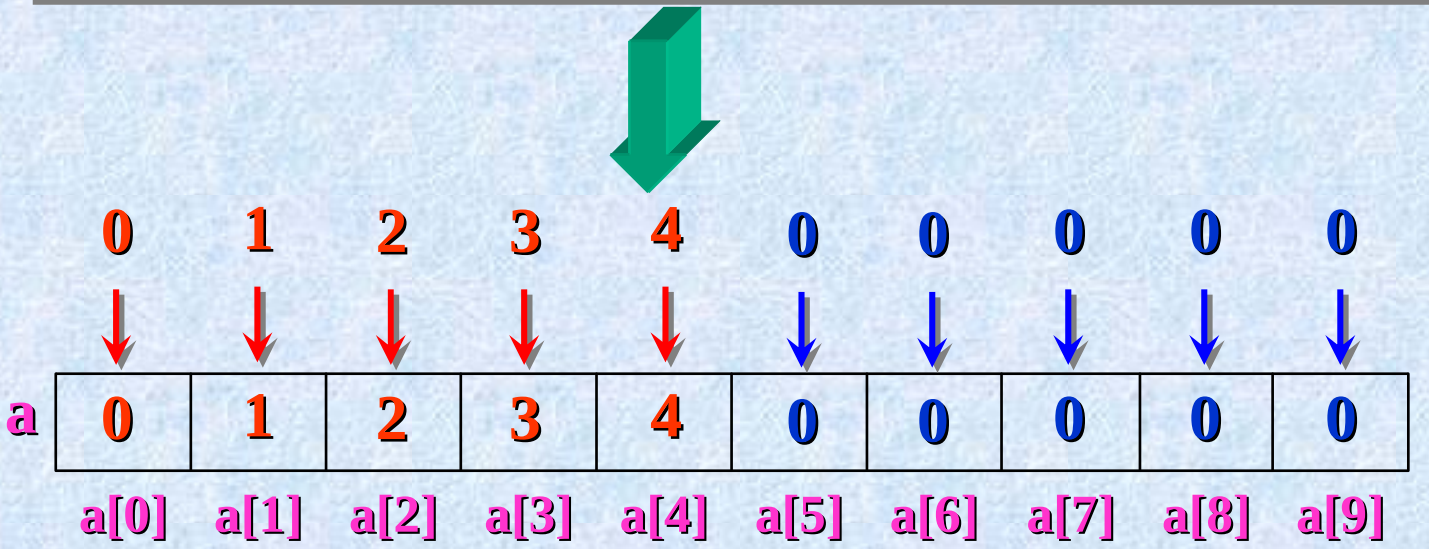
2) define M 10

float B[M];

➢ 初始化赋值说明:

(4) 如果表达式的个数小于数组的大小，则未指定值的数组元素被赋值为0;

例 `int a[10] = {0, 1, 2, 3, 4};`



(5) 当对全部数组元素赋初值时，可以省略数组变量的大小，此时数组变量的实际大小就是初值列表中表达式的个数。

例 `char str[ ] = {'a', 'b', 'c', 'd', 'e' };`
则数组str的实际大小为5。

一维数组初始化

- **int A[5] = { 0,2,4,6,8 };**

  double D[ ] = { 1.2, 2.3, 3.4 };

  int B[8]={1,2};

一维数组赋值

- 在程序中逐个数据元素赋值
- 在程序中通过输入语句赋值

➢ 一维数组在程序中赋值

- 使用循环语句来逐一赋值

例如，将数组a的各元素赋值成奇数序列。

```
int a[10], i;
for (i = 0; i < 10; i++)
    a[i] = 2* i + 1;
```

例如，接受用户键盘输入赋值给数组各元素。

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

判断下列赋值是否正确？

```
int a[3];
scanf ("%d%d%d", a);
```

- 练习：判断正误。

(1) float a[5];

a = {1,2,3,4,5};

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

(2) int a[10], i;

a[0] = 0; a[2] = 2;

for(i=3; i<10; i++)

scanf(“%d”,&a[i]);

scanf(“%d”, &a);

scanf(“%d”, a);

- 使用**memset**函数赋值

  void *memset(void *s, char ch, unsigned n)

  功能：实现对内存块**s**的各个字节单元整体赋同样的值。

  例： **int a[10];**

  memset(a, 0 , 10*sizeof(int));

  char str[100];

  memset(str, ‘a’, 100);

- 使用**memcpy**函数实现数组间的赋值

  void *memcpy(void *d, void *s, unsigned n)

  功能：将以**s**为首地址的连续空间**(n**个字节）的值拷贝给以**d**为首地址的连续空间。

  例：**int b[5],a[5]={1,2,3};**

  memcpy(b,a,5*sizeof(int))

#include <memory.h>

练习

输入一行字符，统计其中各个大写字母出现的次数。

exc91.cpp

4、一维数组应用举例

输入一行字符，统计其中各个大写字母出现的次数。

```
#include <stdio.h>
#include <memory.h>
void main ( void )
{
   char ch;
   int  num[26], i;
   memset (num, 0, 26*sizeof(int));   //初始化数组num
   while ((ch = getchar( )) != '\n')     //输入字符串，判断统计
      if (ch >= 'A' && ch <= 'Z')      //是否为大写字母
         num[ch-'A']++;
   for (i = 0; i < 26; i++)   //输出结果
   {
     if (i % 9 == 0)
        printf ("\n");
     printf ("%c(%d) ", 'A'+i, num[i]);
   }
   printf ("\n");
}
```

运行结果：

```
AABBCCxyYzEEE ↙
A(2) B(2) C(2) D(0) E(3) F(0) G(0) H(0) I(
J(0) K(0) L(0) M(0) N(0) O(0) P(0) Q(0) R
S(0) T(0) U(0) V(0) W(0) X(0) Y(1) Z(0)
```

4、一维数组应用举例

输入一行字符，统计其中各个大写字母出现的次数。

运行结果：

AABBCCxyYzEEE ↙

A(2) B(2) C(2) D(0) E(3) F(0) G(0) H(0) I(0)
J(0) K(0) L(0) M(0) N(0) O(0) P(0) Q(0) R(0)
S(0) T(0) U(0) V(0) W(0) X(0) Y(1) Z(0)

练习

- 冒泡排序，升序

  a[0], a[1], a[2],……, a[N-1]

  从**a[0]**至**a[N-1]**，相邻两个比较，若非正序，交换位置;

  从**a[0]**至**a[N-2]**，相邻两个比较，若非正序，交换位置;

  …

  从**a[0]**至**a[1]**，比较。

exc92.cpp

一维数组应用举例

用冒泡排序法将10个整数按照从小到大的顺序排序

排序过程：

(1) 比较第一个数与第二个数，若为逆序a[0]>a[1]，则交换；然后比较第二个数与第三个数；依次类推，直至第n-1个数和第n个数比较为止——第一趟冒泡排序，结果最大的数被安置在最后一个元素位置上；

(2) 对前n-1个数进行第二趟冒泡排序，结果使次大的数被安置在第n-1个元素位置；

(3) 重复上述过程，共经过n-1趟冒泡排序后，排序结束


38
49
65
76
13
27
30
97
初始关键字
n = 8
38
49
65
13
27
30
76
97
第一趟
38
49
13
27
30
65
76
第二趟
38
13
27
30
49
65
第三趟
13
27
30
38
49
第四趟
13
27
30
38
第五趟
13
27
30
第六趟
13
27
第七趟
冒泡排序图示效果

不足之处：对已排好序的序列仍然要进行9轮冒泡操作，尽管不会有任何数据交换操作。

exc92.cpp

```
#include <stdio.h>
#define  NUM   10
void main ( )
{
  int a[NUM], i, j, t;
  printf ("input %d numbers: \n", NUM);
  for (i = 0; i < NUM; i++)  //输入NUM个整数
     scanf ("%d", &a[i]);
  for (i = 1; i < NUM; i++)          //趟数，共NUM-1趟
    for (j = 0; j < NUM - i; j++)  //实现一次冒泡操作
      if (a[j] > a[j+1])   //交换a[j]和a[j+1]

          t = a[j];
          a[j] = a[j+1];
          a[j+1] = t;
        }
  //输出排好序的数据
  printf ("the sorted nu
  for (i = 0; i < NUM; i++)
     printf ("%d ", a[i]);
}
```

运行结果：

input 10 numbers:

10 1 2 7 6 8 9 3 4 5↙

the sorted numbers:

1 2 3 4 5 6 7 8 9 10

如何修改呢？

对冒泡排序的改进:

当一次冒泡过程中发现没有交换操作时，表明序列已经排好序了，便终止冒泡操作。为了标记在比较过程中是否发生了数据交换，在程序中设立一个标志变量flag，在每趟比较前，把flag变量置为0，如果在这趟比较过程中发生了交换，把变量flag的值置为1。在这一趟比较结束后判断如果flag变量取值等于0表示可以结束排序过程，否则进行下一趟比较。

exc93.cpp

对冒泡排

当一
排好序了
生了数据
较前，把
换，把变
flag变量
趟比较。

exc93.cpp

```
#include <stdio.h>
#define  NUM   10
void main ( )
{
  int a[NUM], i, j, t, flag;
  printf ("input %d numbers: \n", NUM);
  for (i = 0; i < NUM; i++)      //输入NUM个整数
     scanf ("%d", &a[i]);
  for (i = 1; i < NUM; i++)      //轮次，共NUM-1次
  {
     flag = 0;
     for (j = 0; j < NUM - i; j++)  //实现一次冒泡操作
        if (a[j] > a[j+1])              //交换a[j]和a[j+1]
        {   t = a[j];   a[j] = a[j+1];  a[j+1] = t;  flag = 1;  }
     if (flag = = 0)  break;
  }
  printf ("the sorted numbers:\n");  //输出排好序的数据
  for (i = 0; i < NUM; i++)
     printf ("%d ", a[i]);
}
```

4、一维数组应用举例

用选择排序法将10个整数按照从小到大的顺序排序

排序过程：

(1) 首先通过n-1次比较，从n个数中找出最小的， 将它与第一个数交换—第一趟选择排序，结果最小的数被安置在第一个元素位置上；

(2) 再通过n-2次比较，从剩余的n-1个数中找出关键字次小的记录，将它与第二个数交换—第二趟选择排序；

(3) 重复上述过程，共经过n-1趟排序后，排序结束。


k
i = 1 初始: [13 38 65 97 76 49 27]
j j j j j j
k
i = 2 一趟: 13 [27 65 97 76 49 38]
j j j j j
二趟: 13 27 [65 97 76 49 38]
三趟: 13 27 38 [97 76 49 65]
四趟: 13 27 38 49 [76 97 65]
五趟: 13 27 38 49 65 [97 76]
六趟: 13 27 38 49 65 76 [97]
选择排序图示效果

exc94.cpp

```
#include <stdio.h>
void main( )
{
    int  a[11], i, j, k, x;
    printf ("Input 10 numbers: \n");
    for (i = 1; i < 11; i++)
       scanf ("%d", &a[i]);
    printf ("\n");
    for (i = 1; i < 10; i++)
    {
       k = i;
       for (j = i+1; j <= 10; j++)
            if (a[j] < a[k])  k = j;
      if (i != k)
      {  x = a[i]; a[i] = a[k]; a[k] = x; }
    }
    printf("The sorted numbers:\n");
    for (i = 1; i < 11; i++)
            printf ("%d ", a[i]);
}
```

练习

exc95.cpp

用数组求**Fibonacci**数列前**20**个数

$$F1=1 \quad (n=1)$$
$$F2=1 \quad (n=2)$$
$$F_n = F_{n-1} + F_{n-2} \quad (n \geq 3)$$

```
int f[20]={1,1};          ←定义数组
for(i=2;i<20;i++)         ←计算
    f[i]=f[i-2]+f[i-1];
for(i=0;i<20;i++)         ←输出数据
  {  if(i%5==0)  printf("\n");
     printf("%12d",f[i]);
  }
```

练习

exc95.cpp

用数组求Fibonacci数列前20个数

```
#include <stdio.h>
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("%12d",f[i]);
      }
}
```

定义数组

计算

输出数据

7.2 二维数组及多维数组

1、 二维数组的定义

➢定义方式：

数据类型　数组名[常量表达式1][常量表达式2];

行数　列数

元素个数=行数*列数

➢数组元素的存放顺序

- 原因:内存是一维的
- 二维数组： 按行序优先
- 多维数组： 最右下标变化最快

int a[3][2]

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

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

➢二维数组理解

二维数组a是由3个元素组成

例　int a[3][4];

a[0]	a[0][0]	a[0][1]	a[0][2]	a[0][3]
a[1]	a[1][0]	a[1][1]	a[1][2]	a[1][3]
a[2]	a[2][0]	a[2][1]	a[2][2]	a[2][3]

行名

每个元素a[i]由包含4个元素的一维数组组成

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

2、二维数组元素的引用

形式：　数组名[下标1][下标2]

3、二维数组元素的初始化

- 分行初始化：

存储类型符　数据类型　数组变量名[行常量表达式][列常量表达式] = {{第0行初值表}，{第1行初值表}，……，{最后1行初值表}};

对数组元素全部赋值

- 按元素排列顺序初始化

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

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

省掉第一维的大小

例：　int a[ ][3] = { 1, 2, 3, 4 };

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

4、二维数组在程序中赋值

例：通过键盘输入对二维数组a各元素赋值

```
int  i, j, a[2][3];
for (i = 0; i < 2; i++)
   for (j = 0; j < 3; j++)
      scanf ("%d", &a[i][j]);
```

例：调用memset函数把数组a的各元素清0

```
int  a[2][3];
memset(a, 0, 6 * sizeof(int));
```

例：通过memcpy函数将数组a各元素的值复制到数组b的各元素中

```
int  b[2][3];
memcpy(b, a, 6 * sizeof(int));
```

5、二维数组的应用举例

【例1】 输入多个学生多门课程的成绩，分别求每个学生的平均成绩和每门课程的平均成绩。

程序设计思想：

要满足上述程序的要求，必须定义一个二维数组，用来存放学生各门课的成绩。这个数组的每一行表示某个学生的各门课的成绩及其平均成绩，每一列表示某门课的所有学生成绩及该课程的平均成绩。因此，在定义这个学生成绩的二维数组时行数和列数要比学生人数及课程门数多1。成绩数据的输入输出以及每个学生的平均成绩、各门课程的平均成绩的计算方法比较简单。

exc96.cpp

```
#include <stdio.h>
#define NUM_std     5       //定义符号常量学生人数为5
#define NUM_course  4       //定义符号常量课程门数为4
void main ( )
{
  int i, j;
  //定义成绩数组，各元素初值为0
  float score[NUM_std+1][NUM_course+1] = {0};
  for (i = 0; i < NUM_std; i++)
    for (j = 0; j < NUM_course; j++)
    {
      printf ("input the mark of %dth courseof %dth student: ",
                  j+1, i+1);
      scanf ("%f", &score[i][j]);  //输入第i个学生的第j门课的成绩
    }
```

```
for (i = 0; i < NUM_std; i++)
{
  for (j = 0; j < NUM_course; j++)
  {
    score[i][NUM_course] += score[i][j];  //求第i个学生的总成绩
    score[NUM_std][j] += score[i][j];     //求第j门课的总成绩
  }
  score[i][NUM_course] /= NUM_course;  //求第i个人的平均成绩
}
for (j = 0; j < NUM_course; j++)
    score[NUM_std][j] /= NUM_std;        //求第j门课的平均成绩
```

```
  printf (" NO.      C1      C2      C3      C4     AVER\n");
  //输出每个学生的各科成绩和平均成绩
  for (i = 0; i < NUM_std; i++)
  {
    printf ("STU%d\t", i+1);
    for (j = 0; j < NUM_course+1; j++)
        printf ("%6.1f\t", score[i][j]);
    printf ("\n");
  }
  printf ("-----------------------------------");  //输出1条短划线
  printf ("\nAVER_C  ");
  for (j = 0; j < NUM_course; j++)    //输出每门课程的平均成绩
    printf ("%6.1f\t", score[NUM_std][j]);
  printf ("\n");
}
```

练习

思考

将一个二维数组行和列元素互换，存到另一个二维数组中。

$$a=\begin{pmatrix}1 & 2 & 3\\ 4 & 5 & 6\end{pmatrix},\quad b=\begin{pmatrix}1 & 4\\ 2 & 5\\ 3 & 6\end{pmatrix}$$

exc97.cpp

```
#include<stdio.h>
 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”);
     }
}
```

程序运行情况

运行结果如下：

```
array a:
    1   2   3
    4   5   6
array b:
    1   4
    2   5
    3   6
```

提问

1、 (1) array是一个一维整形数组, 有10个元素, 前6个元素的初值是9, 4, 7, 49, 32, -5, 请写出正确的说明语句。

```
int array[10] = {9,4,7,49,32,-5};
```

(2) 如何用scanf函数输入数组的第二个元素。

```
scanf("%d",&array[1]);
```

2、 定义一个三行四列的浮点型数组score

```
float  score[3][4];
```

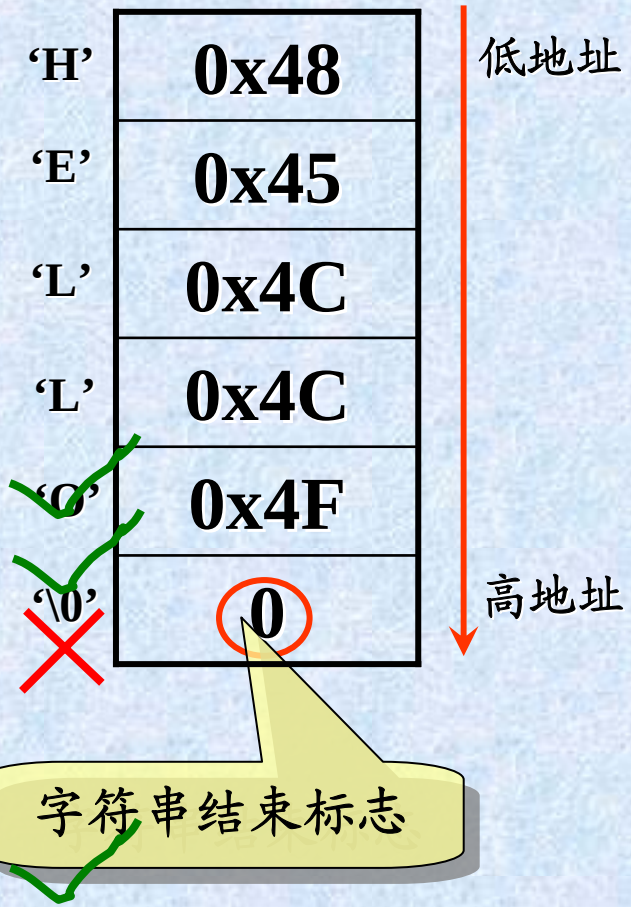
思考 **如何存放字符串？**

字符串与数组

1、字符串的本质

字符串是一种以‘\0’结尾的字符数组。

如：字符串常量"HELLO"的内存映像



2、字符数组

- 定义
- 字符数组的初始化
  - 逐个字符赋值
  - 用字符串常量

例： char c[10], ch[3][4];

3、字符及字符串操作的常用函数

➢ 字符串的输入

● gets函数

格式：gets(字符数组)　　//应包含的.h文件为stdio.h

功能：从键盘输入一以回车结束的字符串放入字符数组中，并自动加'\0'

说明：输入串长度应小于字符数组维数

例：
```
char  str[80];
gets (str);
```
当输入：I□love□china!↙（□表示空格，↙表示回车）时，str中的字符串将是："I love china!"

scanf函数的使用：

例：利用scanf函数可以连续输入多个字符串，输入时，字符串间用空格分隔。

```
char  str1[40], str2[40], str[40];
scanf ("%s%s%s", str1, str2, str3);
```

输入：I□love□china!↙

str1："I"，str2："love"，str3："china!"。

例：使用%ns格式控制符 限制输入的字符个数。

```
char  str[10];
scanf ("%9s", str);  //最多可读入9个非空格字符到str中
```

gets	scanf
输入的字符串中可包含空格字符	输入的字符串中不可包含空格字符
只能输入一个字符串	可连续输入多个字符串（使用%s%s...）
不可限定字符串的长度	可限定字符串的长度（使用%ns）
遇到回车符结束	遇到空格符或回车符结束

3、字符及字符串操作的常用函数

➢ 字符串的输出

- **puts函数**

格式：puts(字符串地址)　　//应包含的.h文件为stdio.h
功能：向显示器输出字符串（输出完，换行）
说明：如果是字符数组，则必须以'\0'结束

- **printf函数**

格式：printf("%s", 字符串地址) //应包含的.h文件为stdio.h
功能：依次输出字符串中的每个字符直到遇到字符'\0'
('\0'不会被输出)

例:

```
char  name[ ] = "John Smith";
printf ("The name is: %s\n", name);
printf ("Last name is: %s\n", &name[5]);
printf ("First name is: %s\n", "John");
```

字符串的输入输出

```
例    用%s
 main()
{    char   str[15];
     scanf("%s", str);
     printf("%s", str);
}
```

数组名表示数组首地址
前面不要加&
输入串长度<数组长度
遇空格或回车结束
自动加'\0'

用字符数组名,
遇'\0'结束

&str[0] <=> str

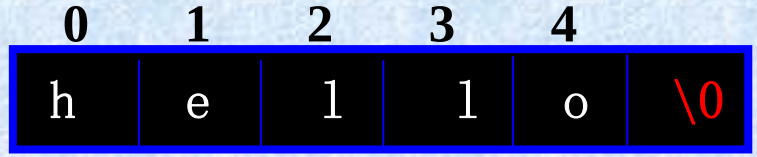
练习

```
1、  main( )
     {  char a[5]={'H','e','l','l','o'};
        printf("%s",a);
     }
```

结果：Hello#-=*

用"%s"输出时，遇'\0'结束

```
2、  main( )
     {  char a[ ]="Hello";
        printf("%s",a);
     }
```



结果：Hello

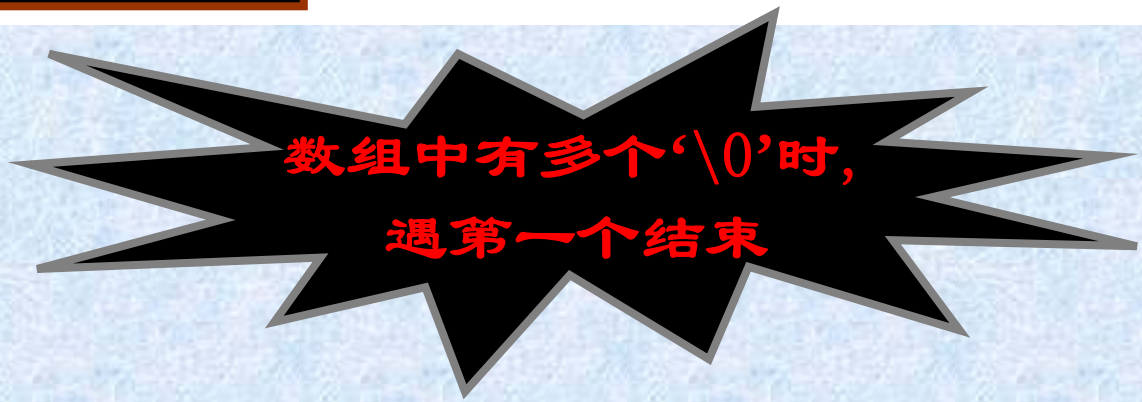
练习

3、 main()

```
{
    char a[]={'h','e','l','\0','l','o','\0'};
    printf("%s",a);
}
```

输出：hel

h	e	l	\0	l	o	\0

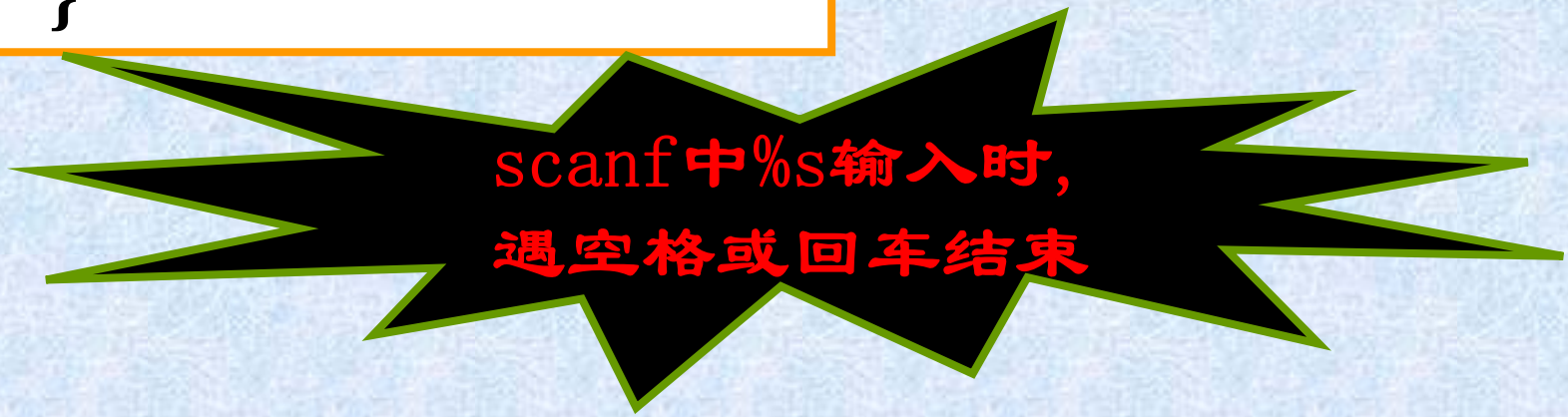


练习

```
4、#include <stdio.h>
   main()
  {  char a[15];
     scanf("%s",a);
     printf("a=%s\n",a);
  }
```

运行情况：
输入：How are you?

输出： a=How



练习

```
#include <stdio.h>
main()
{   char a[15],b[5],c[5];
    scanf("%s%s%s",a,b,c);
    printf("a=%s\nb=%s\nc=%s\n",a,b,c);
}
```

输入：How are you?
输出：a= How
b= are
c= you?

思考

例　若准备将字符串“This is a string.”记录下来，错误的输入语句为：

（A）scanf(“%20s”,s);

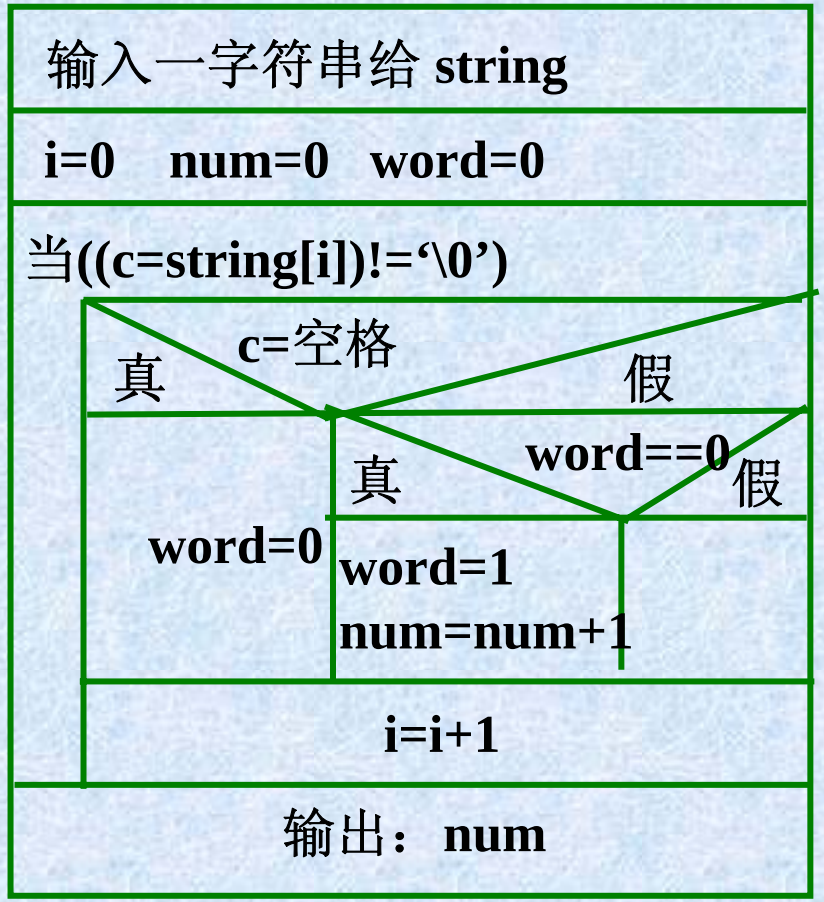
（B）for(k=0;k<18;k++)
　　s[k]=getchar();

（C）while((c=getchar())!=‘\n’)
　　s[k++]=c;

练习

exc98.cpp

输入一行字符，统计其中有多少个单词



练习

exc98.cpp

```
#include <stdio.h>
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 the line\n",num);
}
```

运行结果：

```
I am a student ↙
There are 4 words in the line
```

练习

exc99.cpp

- 编程，找鞍点。
- 输入二维矩阵，查找该矩阵鞍点的位置，该位置上的元素在该行上最大，在该列上最小。如果有，输出其所在的行、列号，如果没有，则输出提示信息。

小结

- 数组的概念——数据集合
- 定义——类型、数组名、数组维数
- 初始化——一般形式和缺省形式
- 元素引用——数组名和下标
- 编程方法——排序、交换、插入、查找
- 常用字符串函数——gets puts strcpy

希望大家能学出好成绩，我们一起努力！

谢谢大家！