

# 第6章 利用数组处理批量数据

- 前几章使用的变量都属于基本类型，例如整型、字符型、浮点型数据，这些都是简单的数据类型。
- 对于有些数据，只用简单的数据类型是不够的，难以反映出数据的特点，也难以有效地进行处理。
- 比如，求两个整数的最大值，定义2个变量存储这两个整数即可；求三个整数的最大值，只需要定义3个变量；…；依次类推，求100个整数的最大值就需要定义100个变量；1000个呢？1000000个呢？显然，不能定义100个、1000个、1000000个变量！
- 同样，多个数的排序也是这样。

- 对于很多数据的情况，在数学上，我们可以用如下序列来表示：

$$a_0, a_1, a_2, \dots, a_n, \dots$$

- 数学上，上述序列可以用数列  $\{ a_n \}$  来表示
- 再如，Fibonacci数列  $\{ F_n \}$ ，在数学上是如下定义的一组整数序列：  $(n=0, 1, 2, 3, \dots)$

$$0, 1, 1, 2, 3, 5, 8, \dots$$

- 下标用来确定数列中元素的序号，也就唯一确定了数列的元素

- C语言中，用数组来表示这样的序列，如用  $a[100]$ ,  $F[n]$  来表示
  - ◆ 数组是一组有序数据的集合，数组中数据的排列有先后次序，下标代表数据在数组中的序号。
  - ◆ 数组名和下标唯一确定数组中的元素
  - ◆ 数组中的元素都属于同一种数据类型
  - ◆ 数组中元素的个数要是确定的
- 定义数组需要如下信息：
  - ◆ 数据类型：标识数组中元素的数据类型
  - ◆ 数组名：用来识别数组
  - ◆ 数组长度：标识数组中元素的个数

## **6.1 一维数组**

## **6.2 二维数组**

## **6.3 字符数组**

# 6.1 一维数组

**6.1.1 怎样定义一维数组**

**6.1.2 怎样引用一维数组元素**

**6.1.3 一维数组的初始化**

**6.1.4 一维数组程序举例**

## 6.1.1 怎样定义一维数组

➤ 一维数组是数组中最简单的，其元素仅需数组名加一个下标，就能惟一确定

◆ 数组必须先定义，再使用

➤ 定义一维数组的一般形式为：

◆ 数据类型符 数组名[整型表达式];

◆ 数组名的命名规则和变量名相同

◆ 如 `int a[10];`

数组元素的类型

数组名

数组长度

逻辑上占一块连续的内存按顺序依次存储，数组名就是数组元素在内存中的首地址

<code>a[0]</code>	<code>a[1]</code>	<code>a[2]</code>	<code>a[3]</code>	<code>...</code>	<code>a[7]</code>	<code>a[8]</code>	<code>a[9]</code>
-------------------	-------------------	-------------------	-------------------	------------------	-------------------	-------------------	-------------------

## 6.1.1 怎样定义一维数组

➤ 定义一维数组的一般形式为：

数据类型符 数组名[整型表达式];

若整型表达式为常量表达式，则其值必须大于**0**

若整型表达式包含变量，则其值必须是确定的

**int a[4+6]; //合法**

**int m=3, n=10;    int a[m+n];    //合法**

**int a[-3+2], b[-1];    //不合法**



## 6.1.2 怎样引用一维数组元素

- 在定义数组并对其中各元素赋值后，就可以引用数组中的元素

- ◆注意：一次只能引用数组中的某一个元素（通过下标），而不能一次引用整个数组全部元素的值

- 引用数组元素的表示形式为：

- ◆数组名 [下标]

- ◆下标可以是整型常量、整型变量或任意整型表达式

- ◆下标的值必须为非负整数，且在有效范围内，C语言编译器不检查下标越界的情况

如 **a[0]=a[5]+a[7]-a[2\*3];** //合法

**int n=5,a[10];**

**a[n]=20;**

合法

## 6.1.2 怎样引用一维数组元素

例**6.1** 对**10**个数组元素依次赋值为**0, 1, 2, 3, 4, 5, 6, 7, 8, 9**，要求按逆序输出。

➤ 解题思路：

- ◆ 定义一个长度为**10**的数组，数组定义为整型
- ◆ 要赋的值是从**0**到**9**，可以用循环来赋值
- ◆ 用循环按下标从大到小输出这**10**个元素

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

```
int main()
```

```
{ int i,a[10];
```

```
  for (i=0; i<10; i++)  
    a[i]=i;
```

```
  for(i=9; i>=0; i--)  
    printf("%d ",a[i]);
```

```
  printf("\n");
```

```
  return 0;
```

```
}
```

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

0	1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---	---

不建议写成i<=9

使a[0]~a[9]的值为0~9

先输出a[9]，最后输出a[0]

9 8 7 6 5 4 3 2 1 0

## 6.1.3 一维数组的初始化

➤ 在定义数组的同时，给各数组元素赋值

◆ 全部元素初始化

● `int a[10]={0,1,2,3,4,5,6,7,8,9};`

◆ 部分元素初始化

● `int a[10]={0,1,2,3,4};` 相当于

● `int a[10]={0,1,2,3,4,0,0,0,0,0};`

◆ 所有元素全部初始化为0

● `int a[10]={0,0,0,0,0,0,0,0,0,0};` 相当于

● `int a[10]={0};`

◆ 定义数组时若为全部元素初始化，可以省略数组长度

● `int a[5]={1,2,3,4,5};` 可写为

● `int a[ ]={1,2,3,4,5};`

## 6.1.4一维数组程序举例

例**6.2** 用数组处理求**Fibonacci**数列问题

➤ 解题思路:

- ◆ 例**5.8**中用简单变量处理的，缺点不能在内存中保存这些数。假如想直接输出数列中第**12**个数，是很困难的。
- ◆ 如果用数组处理，每一个数组元素代表数列中的一个数，依次求出各数并存放在相应的数组元素中

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

```
int main()
```

```
{ int i; int f[20]={0,1};
```

```
for(i=2;i<20;i++)  
    f[i]=f[i-2]+f[i-1];
```

```
for(i=0;i<20;i++)
```

```
{ if(i>0&& i%5==0) printf("\n");
```

```
    printf("%12d",f[i]);
```

```
}
```

初始化Fib数组的  
前2个元素

完成计算Fib数组  
的前20个元素

输出Fib数列的前20个  
元素，每5个数一行

0	1	1	2	3
5	8	13	21	34
55	89	144	233	377
610	987	1597	2584	4181

**例6.3** 有**10**个地区的面积，要求对它们按由小到大的顺序排列。

➤ 解题思路：

◆排序的规律有两种：一种是“升序”，从小到大；另一种是“降序”，从大到小

◆把题目抽象为：“对**n**个数按升序排序”

◆默认排序方式：按照升序排序

◆必须掌握的排序方法

●冒泡排序、选择排序、插入排序

◆冒泡排序算法

```
for(i=0;i<5;i++)  
    if (a[i]>a[i+1])  
    { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

a[0]	9	8	8	8	8	8
a[1]	8	9	5	5	5	5
a[2]	5	5	9	4	4	4
a[3]	4	4	4	9	2	2
a[4]	2	2	2	2	9	0
a[5]	0	0	0	0	0	9

大数下沉，小数上升



```
for(i=0;i<4;i++)  
  if (a[i]>a[i+1])  
  { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

a[0]	8	5	5	5	5
a[1]	5	8	4	4	4
a[2]	4	4	8	2	2
a[3]	2	2	2	8	0
a[4]	0	0	0	0	8
a[5]	9	9	9	9	9

```
for(i=0;i<3;i++)  
    if (a[i]>a[i+1])  
    { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

a[0]	5	4	4	4
a[1]	4	5	2	2
a[2]	2	2	5	0
a[3]	0	0	0	5
a[4]	8	8	8	8
a[5]	9	9	9	9

```
for(i=0;i<2;i++)  
    if (a[i]>a[i+1])  
    { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

a[0]	4	2	2
a[1]	2	4	0
a[2]	0	0	4
a[3]	5	5	5
a[4]	8	8	8
a[5]	9	9	9

```
for(i=0;i<1;i++)  
    if (a[i]>a[i+1])  
    { t=a[i];a[i]=a[i+1];a[i+1]=t; }
```

a[0]	2	0
a[1]	0	2
a[2]	4	4
a[3]	5	5
a[4]	8	8
a[5]	9	9

```
for(i=0;i<5;i++)  
    if (a[i]>a[i+1])  
    { ..... }
```

```
for(i=0;i<4;i++)  
    if (a[i]>a[i+1])  
    { ..... }
```

.....

```
for(i=0;i<1;i++)  
    if (a[i]>a[i+1])  
    { ..... }
```

The diagram illustrates the iteration counts for nested loops. It shows the code: `for(j=0;j<5;j++)` followed by `for(i=0;i<5-j;i++)`. A red box highlights `j<5` in the first loop, with a callout bubble stating "n个数 j<n-1". Another red box highlights `i<5-j` in the second loop, with a callout bubble stating "n个数 i<n-1-j".

```
for(j=0;j<5;j++)  
    for(i=0;i<5-j;i++)  
        if (a[i]>a[i+1])  
        { ..... }
```

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

```
int main()
```

```
{
```

```
    int a[10], i, j;
```

```
    printf("Please input 10 numbers:\n");
```

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

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

```
    printf("\n");
```

```
    for(i=0; i<9; i++)
```

```
        for(j=0; j<9-i; j++)
```

```
            if(a[j]>a[j+1]){
```

```
                int t=a[j]; a[j]=a[j+1]; a[j+1]=t; }
```

冒泡排序核心代码

```
Please input 10 numbers:
```

```
23 54 35 78 231 123 56 87 69 101
```

```
The sorted numbers:
```

```
23 35 54 56 69 78 87 101 123 231
```

```
}
```

## 6.2 二维数组

	队员1	队员2	队员3	队员4	队员5	队员6
1分队	2456	1847	1243	1600	2346	2757
2分队	3045	2018	1725	2020	2458	1436
3分队	1427	1175	1046	1976	1477	2018

```
float pay[3][6];
```

## 6.2 二维数组

6.2.1 怎样定义二维数组

6.2.2 怎样引用二维数组的元素

6.2.3 二维数组的初始化

6.2.4 二维数组程序举例



## 6.2.1 怎样定义二维数组

- 二维数组定义的一般形式为  
数据类型符 数组名[整型表达式][整型表达式];  
如: **float a[3][4], b[5][10];**
- 二维数组可被看作是一种特殊的一维数组:  
它的元素又是一个一维数组
- 例如, 把**a**看作是一个一维数组, 它有**3**个元素:  
**a[0]、a[1]、a[2]**
- 每个元素又是一个包含**4**个元素的一维数组

逻辑上占一块  
连续的内存,  
按行序依次存储

<b>a[0]</b>	<b>a[0][0]</b>	<b>a[0][1]</b>	<b>a[0][2]</b>	<b>a[0][3]</b>
<b>a[1]</b>	<b>a[1][0]</b>	<b>a[1][1]</b>	<b>a[1][2]</b>	<b>a[1][3]</b>
<b>a[2]</b>	<b>a[2][0]</b>	<b>a[2][1]</b>	<b>a[2][2]</b>	<b>a[2][3]</b>

## 6.2.2怎样引用二维数组的元素

➤ 同一维数组相同，只能通过下标引用二维数组的某个元素，而不能一次引用整个数组。

➤ 引用二维数组元素的表示形式为：

数组名 [行下标] [列下标]

行下标和列下标都必须是整型表达式，其值不能为负

➤ **b[1][2]=a[2][3]/2**    合法

➤ **for(i=0;i<m;i++)**

**printf("%d,%d\n",a[i][0],a[0][i]);**    合法

## 6.2.3 二维数组的初始化

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

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

`int a[3][4]={ {1},{5},{9}};`等价于

`int a[3][4]={ {1,0,0,0},{5,0,0,0},{9,0,0,0}};`

`int a[3][4]={ {1},{5,6}};`相当于

`int a[3][4]={ {1},{5,6},{0}};`

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

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

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

## 6.2.4 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。

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

➤ 解题思路：

- ◆ 可以定义两个数组：数组**a**为**2**行**3**列，存放指定的**6**个数
- ◆ 数组**b**为**3**行**2**列，开始时未赋值
- ◆ 将**a**数组中的元素**a[i][j]**存放到**b**数组中的**b[j][i]**元素中
- ◆ 用嵌套的**for**循环完成

```

#include <stdio.h>
int main()
{
    int a[2][3] = {{1,2,3},{4,5,6}}, b[3][2], i, j;
    printf("Array a:\n");
    for(i=0; i<2; i++){ 处理a的一行中各元素
        for(j=0; j<3; j++){ 处理a中某一行元素
            printf("%-6d", a[i][j]); 输出a的各元素
        }
        printf("\n");
    }
    printf("Array b:\n");
    for(i=0; i<3; i++){
        for(j=0; j<2; j++){
            b[i][j] = a[j][i]; a元素值
            printf("%-6d", b[i][j]);
        }
        printf("\n");
    }
    return 0;
}

```

```

Array a:
1      2      3
4      5      6
Array b:
1      4
2      5
3      6

```

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。

➤ 解题思路：采用“打擂台算法”

◆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

◆第**3**人与台上的人比武，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为止，最后留在台上的是冠军

➤ 方法：

◆先把**a[0][0]**的值赋给变量**max**，记录行下标和列下标

◆**max**用来存放当前已知的最大值

◆**a[0][1]**与**max**比较，如果**a[0][1]>max**，则表示**a[0][1]**是已经比过的数据中值最大的，把它的值赋给**max**，取代了**max**的原值，重新记录行下标和列下标

◆以后依此处理，最后**max**就是最大的值

**max=a[0][0],row=0,col=0**

**for i=0 to 2**

**for j=0 to 3**

**a[i][j]>max**

真

假

**max=a[i][j]**

**row=i**

**col=j**

**输出: max,row,col**

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

```
int main()
```

```
{
```

```
    int i,j,row,col,max;
```

```
    int a[3][4]={ {1,2,3,4}, {9,8,7,6}, {-10,10,-5,2} };
```

```
    max=a[0][0]; row=0; col=0;
```

```
    for (i=0;i<3;i++)
```

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

```
            if (a[i][j]>max)
```

```
                { max=a[i][j]; row=i; col=j; }
```

```
    printf("max=%d\nrow=%d\n  
        column=%d\n",max,row,col);
```

```
    return 0;
```

```
}
```

记最大值

记行号

记列号

```
max=10  
row=2  
column=1
```