第5章

批量数据处理——数组

5.1 知识点回顾

5.1.1 一维数组

一维数组是一个有序数据的集合，定义了一个一维数组就是定义了一组同类型的变量。定义一个一维数组要说明3个问题：第一，数组是一个变量，应该有一个变量名；第二，数组中有多少个元素；第三，每个元素的数据类型。综合上述3点，C++中一维数组的定义方式如下：

```
类型名  数组名[常量表达式];
```

其中，类型名指出了数组元素的数据类型；数组名是存储该数组的变量名；常量表达式指出数组中元素的个数。数组定义特别要注意的是数组中元素的个数是用一个常量表达式说明的，即元素个数在写程序时就已经确定。

定义数组就是定义了一块连续的空间，空间的大小等于元素的个数乘以每个元素所占的空间大小。数组元素按序存放在这块空间中。

数组的使用一般是引用它的元素，数组元素的表示方式：数组名[下标]，下标表示元素的编号。C++的下标从0开始。“数组名[下标]”被称为下标变量。下标变量中的下标可以是常量、变量或任何计算结果为整型数的表达式。因此，访问数组的所有成员可以用一个for语句实现。

与其他变量一样，可以在定义数组时为数组元素赋初值。数组有一组初值，这组初值用一对大括号括起来，值与值之间用逗号分隔。数组的初始化可用以下3种方法实现。

(1) 对所有数组元素赋初值。例如：

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

表示将0、1、2、3、4、5、6、7、8、9依次作为$a[0]$、$a[1]$、$a[2]$、$a[3]$、$a[4]$、$a[5]$、$a[6]$、$a[7]$、$a[8]$、$a[9]$的初值。

(2) 可以对数组的一部分元素赋值。例如：

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

表示数组 a 的前 5 个元素的值分别是 0、1、2、3、4，后 5 个元素的值为 0。在对数组元素赋初值时，总是按从下标小的元素到下标大的元素的次序，没有得到初值的元素的初值为 0。因此，需要将数组的所有元素的初值都设为 0，可简单地写成：

```
int a[10] ={0};
```

(3) 在对全部数组元素赋初值时，可以不指定数组大小，C++ 会根据给出的初值的个数确定数组的规模。例如：

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

表示 a 数组有 10 个元素，它们的初值分别为 0、1、2、3、4、5、6、7、8、9。

C++ 11 中，数组还可以用范围 for 访问，只需要用数组名替代列表。例如，a 是一个整型数组，输出数组 a 的所有元素可以用语句

```
for(int x: a) cout <<x <<'\t';
```

或

```
for (auto x: a) cout <<x <<'\t';
```

5.1.2　二维数组

数组的元素可以是任何类型。如果数组的每一个元素又是数组，则称为多维数组。最常用的多维数组是二维数组，即每一个元素都是一个一维数组的一维数组。

二维数组可以看成数学中的矩阵，它由行和列组成。定义一个二维数组必须说明它有几行几列。二维数组定义的一般形式如下：

```
类型名　数组名[常量表达式 1][常量表达式 2];
```

类型名是二维数组中每个元素的类型，常量表达式 1 给出二维数组的行数，常量表达式 2 给出了二维数组的列数。当把二维数组看成是元素为一维数组的数组时，也可以把常量表达式 1 看成是一维数组的元素个数，常量表达式 2 是每个元素(也是一个一维数组)中元素的个数。例如：

```
int  a[4][5];
```

表示定义了一个 4 行、5 列的矩阵，矩阵的每个元素是整型。也可以看成定义了一个有 4 个元素的一维数组，每个元素的类型是一个由 5 个元素组成的一维数组，$a[i]$表示第 i 行。

一旦定义了数组 a，就相当于定义了 20 个整型变量，即 $a[0][0]$、$a[0][1]$、…、$a[0][4]$、…、$a[3][0]$、$a[3][1]$、…、$a[3][4]$。第一个下标表示行号，第二个下标表示列号。例如，$a[2][3]$是数组 a 的第二行第三列的元素。同一维数组一样，下标的编号也是从 0 开始的。

二维数组的初始化有以下 3 种方法。

(1) 对所有的元素赋初值。例如：

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

编译器依次把大括号中的值赋给第一行的每个元素,然后是第二行的每个元素,以此类推。初始化后的数组元素如下:

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

可以通过大括号把每一行括起来使这种初始化方法表示得更加清晰。

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

(2) 对部分元素赋值。同一维数组一样,二维数组也可以对部分元素赋值。C++ 将初始化列表中的数值按行序依次赋给每个元素,没有赋到初值的元素初值为 0。例如:

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

初始化后的数组元素如下:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(3) 对每一行的部分元素赋初值。例如:

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

初始化后的数组元素如下:

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 5 & 0 & 0 & 0 \end{bmatrix}$$

一旦定义了一个二维数组,系统就在内存中准备了一块连续的空间,数组的所有元素都存放在这块空间中。在内存中,二维数组的元素是按行序存放的,即先放第一行的元素,然后放第二行的元素。

5.1.3 字符串

除了科学计算以外,计算机最主要的用途就是文字处理。文字处理的基本单元是一个单词或一个句子。无论是单词还是句子都由一系列字符组成。由一系列字符组成的一个单元称为字符串。C++ 中,字符串常量是用一对双引号括起来、由'\0'作为结束符的一组字符。但 C++ 语言并没有字符串这样一个内置类型,字符串是被当作一个字符数组来保存。字符之间的次序由数组元素的位置来表示。如要将字符串"Hello,world"保存在一个数组中,这个数组的长度至少为 12 个字符。可以用下列语句将"Hello,world"保存在字符数组 ch 中:

```
char ch[ ] ={ 'H', 'e', 'l', 'l', 'o', ',', 'w', 'o', 'r', 'l', 'd', '\0'};
```

C++ 分配一个 12 个字符的数组,将这些字符存放进去。在定义数组时也可以指定

长度，但此时要记住数组的长度是字符个数加 1，因为还有结束符'\0'。

对字符串赋初值，C++ 还提供了另外两种简单的方式：

```
char ch[ ] ={"Hello,world"};
```

或

```
char ch [ ] ="Hello,world";
```

这两种方法等价。C++ 都会定义一个 12 个字符的数组，把这些字符依次放进去，最后插入'\0'。

由于字符串的本质是一个字符数组，所以不能对字符串整体进行操作。例如，对字符串进行复制、比较等，需要像数组一样通过对数组元素的操作实现各种字符串的操作。但在大多数场合，字符串确实是一个整体。为了解决这个问题，C++ 提供了一个专门处理字符串的函数库，又在输入输出中对字符串做了特殊处理。

字符串的输入输出有下面 3 种方法。

(1) 逐个字符的输入输出，这种做法与普通的数组操作一样。

(2) 将整个字符串一次性地用“>>”和“<<”输入或输出。

(3) 通过 cin 对象的成员函数 getline 输入。

如果定义了一个字符数组 ch，输入一个字符串放在 ch 中可直接用语句：

```
cin >>ch;
```

输出 ch 的内容可直接用语句：

```
cout <<ch;
```

用“>>”输入时，要注意输入的字符串的长度不能超过数组的长度。“>>”操作不检查数组的长度，如果输入的字符串超过数组长度，就会发生内存溢出，会出现一些无法预知的错误。因此，用“>>”直接输入字符串时，最好在输出的提示信息中告知用户允许的最长字符串长度。

与其他类型一样，“>>”输入是以空格符、回车符或制表符作为结束符的，所以输入的字符串中不能包含这些字符。因此当一个字符串中真正包含一个空格时将无法输入，此时可用 cin 的成员函数 getline 实现。getline 函数的格式：

```
cin.getline(字符数组, 数组长度, 结束标记);
```

它从键盘接收一个包含任意字符的字符串，直到遇到指定的结束标记或到达数组长度减 1（因为字符串的最后一个字符必须是'\0'，所以必须为'\0'预留空间）。结束标记也可以不指定，此时默认回车符为结束标记。例如，ch1 和 ch2 都是长度为 80 的字符数组，执行语句：

```
cin.getline(ch1, 80, '.');
cin.getline(ch2, 80);
```

如果对应的输入为 aaa bbb ccc. ddd eee fff ggg↙，则 ch1 的值为"aaa bbb ccc"，ch2 的值为"ddd eee fff ggg"。

处理字符串的函数在库 cstring 中。使用这些函数的程序必须包含头文件 cstring。库 cstring 包含的主要的字符串处理函数如表 5-1 所示。

表 5-1 主要的字符串处理函数

函 数	作 用
strcpy(dst, src)	将字符串从 src 复制到 dst。函数的返回值是 dst 的地址
strncpy(dst, src, n)	至多从 src 复制 n 个字符到 dst。函数的返回值是 dst 的地址
strcat(dst, src)	将 src 拼接到 dst 后。函数的返回值是 dst 的地址
strncat(dst, src, n)	从 src 至多取 n 个字符拼接到 dst 后。函数的返回值是 dst 的地址
strlen(s)	返回字符串 s 的长度,即字符串中的字符个数
strcmp(s1, s2)	比较 s1 和 s2。如果 s1>s2 返回值为正数,s1=s2 返回值为 0,s1<s2 返回值为负数
strncmp(s1, s2, n)	与 strcmp 类似,但至多比较 n 个字符
strchr(s, ch)	返回一个指向 s 中第一次出现字符 ch 的地址
strrchr(s, ch)	返回一个指向 s 中最后一次出现字符 ch 的地址
strstr(s1, s2)	返回一个指向 s1 中第一次出现字符串 s2 的地址

5.2 习题解答

5.2.1 简答题

1. 数组的两个特性是什么?

【解】数组的第一个特性是所有数组元素的类型是相同的,第二个特性是数组元素之间是有序的。

2. 写出以下数组变量的定义。

(1) 一个含有 100 个浮点型数据的名为 realArray 的数组。

(2) 一个含有 16 个布尔型数据的名为 inUse 的数组。

(3) 一个含有 1000 个字符串,每个字符串的最大长度为 20 的名为 lines 的数组。

【解】

```
(1) double  realArray[100];
(2) bool  inUse[16];
(3) char  lines[1000][21];
```

3. 用 for 循环实现下述整型数组的初始化操作。

squares

0	1	4	9	16	25	36	49	64	81	100
0	1	2	3	4	5	6	7	8	9	10

【解】在数组 squares 中，每个元素的值正好是对应下标的平方，因此可用语句：

```
for (int i =0; i <=10; ++i) squares[i] =i * i;
```

4. 用 for 循环实现下述字符型数组的初始化操作。

array

a	b	c	d	e	f	…	w	x	y	z
0	1	2	3	4	5	…	22	23	24	25

【解】观察数组 array 的元素值，发现第 i 个元素值是第 i 个小写字母，元素值与元素下标的关系是'a'＋i，所以初始化数组 array 可以用语句：

```
for(i =0; i <26; ++i)  array[i] ='a' +i;
```

5. 什么是数组的配置长度和有效长度？

【解】有时在编写程序时无法确定所要处理的数据量，因此无法确定数组的大小。这时可以按可能的最大的数据量定义数组。定义数组时给定的数组规模称为配置长度，在执行时真正存入数组中的元素个数称为有效长度。

6. 什么是多维数组？

【解】数组元素本身又是一个数组的数组称为多维数组。

7. 要使整型数组 $a[10]$ 的第一个元素值为 1，第二个元素值为 2，……，最后一个元素值为 10，某程序员写了下面语句，请指出错误。

```
for (i =1; i <=10; ++i) a[i] =i;
```

【解】数组的下标是从 0 开始，10 个元素的数组下标的合法范围是 0～9，即 a[0]的值是 1，……，a[9]的值是 10。正确的语句是：

```
for (i =0; i <10; ++i) a[i] =i+1;
```

8. 有定义“char s[10];”执行下列语句会有什么问题？

```
strcpy(s, "hello world");
```

【解】会发生内存溢出。C++ 语言中，字符串必须以'\0'结束，字符串"hello world"本身长度为 11，再加上'\0'应该是 12 个字符，而数组 s 只有 10 字节的空间。

9. 写出定义一个整型二维数组并赋如下初值的语句。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

【解】该数组除了对角线外的其他元素均为 0，对角线元素值为 1、2、3、4。该数组有两种赋初值方法：一种是定义时按行赋初值；另一种是定义时将所有元素赋值为 0，然后为对角线赋值。

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

```
(2) int a[4][4] ={0};
    for (k =0; k <4; ++k) a[k][k] =k+1;
```

10. 定义了一个 26×26 的字符数组,写出为它赋如下值的语句。

a b c d e f … x y z
b c d e f g … y z a
⋮ ⋮
y z a b c d … v w x
z a b c d e … w x y

【解】该问题有 3 种解决方法。

(1) 该数组的第一行是小写字母 a～z,其余每一行都是上一行循环左移一位,即执行“a[i][j]=a[i-1][j+1]”。但还有一个问题是如何实现循环,即如何将第二行的最后一个元素设为 a,如何将第三行的最后一个元素设为 b,依此类推。最简单的方法是对最后一个元素做一个特殊处理。这个方案的实现语句如下:

```
for (i =0; i <26; ++i) ch[0][i] ='a' +i;
for (i =1; i <26; ++i) {
    for (j =0; j <25; ++j)
        ch[i][j]=ch[i-1][j+1];
    ch[i][25] =ch[i-1][0];
}
```

(2) 第二种解决方案是用取模运算将列号 25 转换为 0,即“(j+1)%26”。按照这个方案实现的语句如下:

```
for (i =0; i <26; ++i) ch[0][i] ='a' +i;
for (i =1; i <26; ++i) {
    for (j =0; j <26; ++j)
        ch[i][j]=ch[i-1][(j+1)%26];
}
```

(3) 方法一和方法二都是将第一行单独处理,然后根据上一行生成下一行,但其实没有必要。仔细观察这个数组,可以发现数组元素值与 i+j 有关。当 i+j 小于 25 时,值为 i+j+'a'。i+j 大于 26 时,重新从'a'开始排列。这两种情况可以用一个表达式表示“(i+j) % 26+'a'”。这段语句如下:

```
for (i =0; i <26; ++i) {
    for (j =0; j <26; ++j)
        ch[i][j]='a' +(i+j) %26;
}
```

5.2.2 程序设计题

1. 编写一个程序,计算两个十维向量的和。

【解】本题主要解决的问题是如何存储十维向量。一个十维向量由 10 个分量组成，每个分量是一个实数。这正好满足数组的两个特性：所有元素的类型是相同的，元素之间是有序的。所以每个十维向量可以用一个 10 个元素组成的 double 型数组表示。计算向量和是计算每个分量之和。完整的实现如代码清单 5-1。

代码清单 5-1　计算两个十维向量的和

```
#include <iostream>
using namespace std;

int main()
{
    double a[10], b[10], c[10];
    int i;

    cout <<"请输入第一个向量的 10 个分量:";
    for (i =0; i <10; ++i) cin >>a[i];
    cout <<"请输入第二个向量的 10 个分量:";
    for (i =0; i <10; ++i) cin >>b[i];

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

    cout <<"(" <<a[0];
    for (i =1; i <10; ++i) cout <<", " <<a[i];
    cout <<") +(" <<b[0];
    for (i =1; i <10; ++i) cout <<", " <<b[i];
    cout <<") =(" <<c[0];
    for (i =1; i <10; ++i) cout <<", " <<c[i];
    cout <<") \n" ;

    return 0;
}
```

2. 编写一个程序，计算两个十维向量的数量积。

【解】两个十维向量（x_0，x_1，x_2，x_3，x_4，x_5，x_6，x_7，x_8，x_9）和（y_0，y_1，y_2，y_3，y_4，y_5，y_6，y_7，y_8，y_9）的数量积的计算公式为 $\sum_{i=0}^{9} x_i y_i$，所以只需要一个 for 循环计算 $x_i y_i$ 之和。具体实现如代码清单 5-2。

代码清单 5-2　计算两个十维向量的数量积

```
#include <iostream>
using namespace std;

int main()
{
```

```
    double a[10], b[10], c =0;
    int i;

    cout <<"请输入第一个向量的 10 个分量:";
    for (i =0; i <10; ++i) cin >>a[i];
    cout <<"请输入第二个向量的 10 个分量:";
    for (i =0; i <10; ++i) cin >>b[i];

    for (i =0; i <10; ++i) c +=a[i] * b[i];

    cout <<"(" <<a[0];
    for (i =1; i <10; ++i) cout <<", " <<a[i];
    cout <<") . (" <<b[0];
    for (i =1; i <10; ++i) cout <<", " <<b[i];
    cout <<") =" <<c <<endl;

    return 0;
}
```

3. 编写一个程序,输入一个字符串,输出其中每个字符在字母表中的序号。对于不是英文字母的字符,输出 0。例如,输入为"acbf8g",输出为 1 3 2 6 0 7。

【解】处理字符串可以用两种方法解决:一种方法是输入一个字符处理一个字符;另一种方法是一次性输入一个字符串存入一个字符数组,然后依次处理数组中的每个字符。第一种方法的实现如代码清单 5-3,第二种方法的实现如代码清单 5-4。

代码清单 5-3　输出字符串中每个字符在字母表中的序号(方案一)

```
#include <iostream>
using namespace std;

int main()
{
    char ch;

    while ((ch =cin.get()) !='\n') {
        if (ch >='A' && ch <='Z') ch =ch -'A' +'a';
        if (ch >='a' && ch <='z') cout <<ch -'a' +1 <<"  ";
        else cout <<"0";
    }
    cout <<endl;

    return 0;
}
```

代码清单 5-4　输出字符串中每个字符在字母表中的序号(方案二)

```
#include <iostream>
using namespace std;

int main()
{
    char ch[80];
    int i;

    cin.getline(ch, 80);
    for (i =0; ch[i] !='\0'; ++i) {
        if (ch[i] >='A' && ch[i] <='Z') ch[i] =ch[i] -'A' +'a';
        if (ch[i] >='a' && ch[i] <='z') cout <<ch[i] -'a' +1 <<"  ";
        else cout <<"0";
    }
    cout <<endl;

    return 0;
}
```

4. 编写一个程序,将输入的一个字符串表示的实数转换成 double 型的数值,并输出该数字乘 2 后的结果。如输入的是"123.456",则输出为 246.912。

【解】将字符串表示的整数转换成整型数想必读者已经很熟悉了。将字符串表示的实数转换成 double 型的数值可以分成两步:第一步,先忽略小数点,将其转换成整型数;第二步,根据小数点后的位数 pos 执行 pos 次除以 10 的运算。完整实现如代码清单 5-5。

代码清单 5-5　字符串转实型数

```
#include <iostream>
using namespace std;

int main()
{
    char ch[80];
    int i, pos =0;
    bool flag =false;                                   //是否正在处理小数点后的数值
    double  num =0;

    cin.getline(ch, 80);
    for (i =0; ch[i] !='\0'; ++i) {                     //遍历每个字符
        if (ch[i] >='0' && ch[i] <='9') num =num * 10 +ch[i] -'0';//处理数字
        if (flag) ++pos;                                //统计小数点后的位数
        if (ch[i] =='.') flag =true;                    //遇到小数点
    }
```