第六章　数组 指针与字符串（一）

清华大学　郑 莉

教材：C++语言程序设计（第4版） 郑莉　清华大学出版社
答疑：每周一17:30—19:00，东主楼8区310

目录

数组的定义与使用

- 数组是具有一定顺序关系的若干相同类型变量的集合体，组成数组的变量称为该数组的元素。

数组的定义

类型说明符　数组名[常量表达式] [常量表达式]…… ；

↑

数组名的构成方法与一般变量名相同。

例如：int a[10];
表示a为整型数组，有10个元素：a[0]...a[9]

例如: int a[5][3];
　　表示a为整型二维数组，其中第一维有5个下标（0~4），第二维有3个下标（0~2），数组的元素个数为15，可以用于存放5行3列的整型数据表格。

数组的使用

- 使用数组元素
  - 数组必须先定义，后使用。
  - 可以逐个引用数组元素。
  - 例如：

    a[0]=a[5]+a[7]-a[2*3]

    b[1][2]=a[2][3]/2

例6-1

```
#include <iostream>
using namespace std;
int main() {
  int a[10], b[10];
  for(int i = 0; i < 10; i++) {
    a[i] = i * 2 - 1;
    b[10 - i - 1] = a[i];
  }
  for(int i = 0; i < 10; i++) {
    cout << "a[" << i << "] = " << a[i] << "   ";
    cout << "b[" << I << "] = " << b[i] << endl;
  }
  return 0;
}
```

数组的存储与初始化

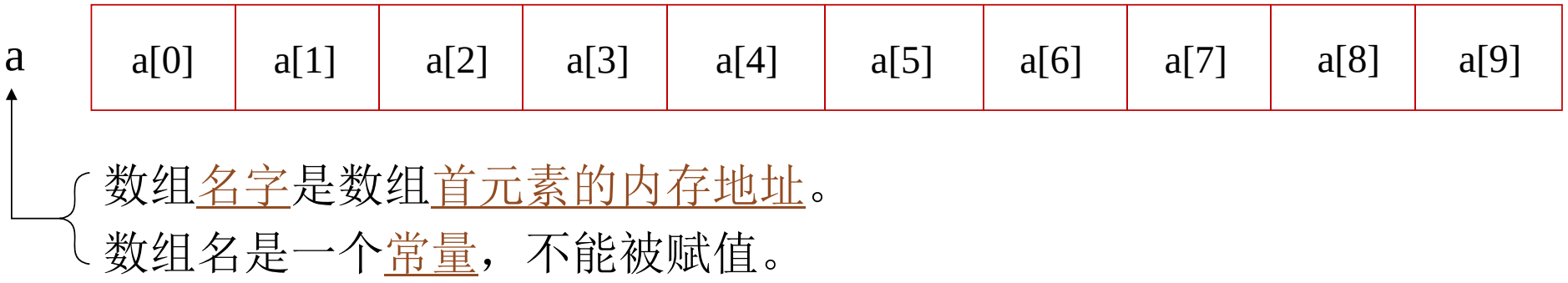
<6.1.2>

一维数组的存储

数组元素在内存中顺次存放，它们的地址是连续的。元素间物理地址上的相邻，对应着逻辑次序上的相邻。

例如：

```
int a[10];
```



数组名字是数组首元素的内存地址。

数组名是一个常量，不能被赋值。

一维数组的初始化

- 列出全部元素的初始值

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

- 可以只给一部分元素指定初值

  例如：int a[10]={0,1,2,3,4};

- 在列出全部数组元素初值时，可以不指定数组长度

  例如：static int a[]={0,1,2,3,4,5,6,7,8,9}

二维数组的存储

- 按行存放

  例如： float a[3][4];

可以理解为：

$$a \begin{cases} a[0] \text{——} a_{00}\ a_{01}\ a_{02}\ a_{03} \\ a[1] \text{——} a_{10}\ a_{11}\ a_{12}\ a_{13} \\ a[2] \text{——} a_{20}\ a_{21}\ a_{22}\ a_{23} \end{cases}$$

其中数组a的存储顺序为：

$$a_{00}\ a_{01}\ a_{02}\ a_{03}\ \ a_{10}\ a_{11}\ a_{12}\ a_{13}\ \ a_{20}\ a_{21}\ a_{22}\ a_{23}$$

二维数组的初始化

- 将所有初值写在一个{}内，按顺序初始化
  - 例如：static int a[3][4]={1,2,3,4,5,6,7,8,9,10,11,12};
- 分行列出二维数组元素的初值
  - 例如：static int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
- 可以只对部分元素初始化
  - 例如：static int a[3][4]={{1},{0,6},{0,0,11}};
- 列出全部初始值时，第1维下标个数可以省略
  - 例如：static int a[][4]={1,2,3,4,5,6,7,8,9,10,11,12};
  - 或：static int a[][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};

- 如果不作任何初始化，局部作用域的非静态数组中会存在垃圾数据，static数组中的数据默认初始化为0
- 如果只对部分元素初始化，剩下的未显式初始化的元素，将自动被初始化为零

单选题 1分

设置

在C++中，下列对二维数组的定义正确的是（ ）。

A int a[5][];

B int a[][5];

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

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

提交

补充6_1: 求Fibonacci数列的前20项，将结果存放于数组中

```
#include <iostream>
using namespace std;
int main() {
  int f[20] = {1,1};          //初始化第0、1个数
  for (int i = 2; i < 20; i++) //求第2～19个数
    f[i] = f[i - 2] + f[i - 1];
  for (i=0;i<20;i++) {    //输出，每行5个数
    if (i % 5 == 0) cout << endl;
     cout.width(12);        //设置输出宽度为12
     cout << f[i];
  }
  return 0;
}
```

运行结果：

```
           1           1           2           3           5
           8          13          21          34          55
          89         144         233         377         610
         987        1597        2584        4181        6765
```

补充6_2: 计算某天是该年的第几天

给出一个包含年月日的日期，输出这个日期在该年中是第几天。要求用一个2*12的数组来存储不同年份的每月的日数，注意输入的判断和闰年的判断。

补充6_2 ：计算某天是该年的第几天

```
#include <iostream>
using namespace std;
int main() {
        int m[2][12] = {{31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
                        {31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}};
        int year, month, day;
        int run = 0;
        cout << "请输入三个正整数作为年月日：";
        cin >> year >> month >> day;
        if (year <= 0 || month < 1 || month > 12) {
                cout << "输入的日期不存在" << endl;
                return 0;
        }
```

补充6_2：计算某天是该年的第几天

```
if ((year % 100 != 0 && year % 4 == 0) || year % 400 == 0)
        run = 1;
if (day < 1 || day > m[run][month-1]) {
        cout << "输入的日期不存在" << endl;
        return 0;
}
int index = 0;
for (int i = 0; i < month - 1; i ++)
        index += m[run][i];
index += day;
cout << "输入日期为一年中的第" << index << "天" << endl;
return 0;
}
```

数组作为函数参数

<6.1.3>

例6-2 使用数组名作为函数参数

主函数中初始化一个二维数组，表示一个矩阵，并将每个元素都输出，然后调用子函数，分别计算每一行的元素之和，将和直接存放在每行的第一个元素中，返回主函数之后输出各行元素的和。

例6-2 使用数组名作为函数参数

```
#include <iostream>
using namespace std;
void rowSum(int a[][4], int nRow) {
  for (int i = 0; i < nRow; i++) {
        for(int j = 1; j < 4; j++)
              a[i][0] += a[i][j];
  }
}
int main() { //主函数
  //定义并初始化数组
  int table[3][4] = {{1, 2, 3, 4}, {2, 3, 4, 5}, {3, 4, 5, 6}};
```

技巧：多维数组通常用多重嵌套循环处理

例6-2 使用数组名作为函数参数

```
  //输出数组元素
   for (int i = 0; i < 3; i++)        {
         for (int j = 0; j < 4; j++)
                cout << table[i][j] << "   ";
         cout << endl;
   }
   rowSum(table, 3);        //调用子函数，计算各行和
  //输出计算结果
   for (int i = 0; i < 3; i++)
         cout << "Sum of row " << i << " is " << table[i][0] << endl;
   return 0;
}
```

运行结果：

```
1   2   3   4
2   3   4   5
3   4   5   6
Sum of row 0 is 10
Sum of row 1 is 14
Sum of row 2 is 18
```

例6_2修改：使用常数组作为函数参数

修改例6_2，以常数组作为函数的参数，观察编译时的情况

```
#include <iostream>
using namespace std;
void rowSum(const int a[][4], int nRow) {
  for (int i = 0; i < nRow; i++) {
        for(int j = 1; j < 4; j++)
              a[i][0] += a[i][j];  //编译错误
  }
}
int main() { //主函数
  //函数体同例7_4，略
}
```

- 数组元素作实参，与单个变量一样。
- 数组名作参数，形、实参数都应是数组名（实质上是地址，关于地址详见6.2），类型要一样，传送的是数组首地址。对形参数组的改变会直接影响到实参数组。

对象数组

<6.1.4 >

对象数组的定义与访问

- 定义对象数组

  类名　数组名[元素个数]；

- 访问对象数组元素

  通过下标访问

  数组名[下标].成员名

对象数组初始化

- 数组中每一个元素对象被创建时，系统都会调用类构造函数初始化该对象。
- 通过初始化列表赋值。

  例：Point a[2]={Point(1,2),Point(3,4)};
- 如果没有为数组元素指定显式初始值，数组元素便使用默认值初始化（调用默认构造函数）。

数组元素所属类的构造函数

- 元素所属的类不声明构造函数，则采用默认构造函数。
- 各元素对象的初值要求为相同的值时，可以声明具有默认形参值的构造函数。
- 各元素对象的初值要求为不同的值时，需要声明带形参的构造函数。
- 当数组中每一个对象被删除时，系统都要调用一次析构函数。

例6-3 对象数组应用举例

```
//Point.h
#ifndef _POINT_H
#define _POINT_H
class Point {    //类的定义
public:                          //外部接口
   Point();
   Point(int x, int y);
   ~Point();
   void move(int newX,int newY);
   int getX() const { return x; }
   int getY() const { return y; }
   static void showCount();     //静态函数成员
private:                         //私有数据成员
   int x, y;
};
#endif//_POINT_H
```

例6-3 对象数组应用举例

```
//Point.cpp
#include <iostream>
#include "Point.h"
using namespace std;
Point::Point() : x(0), y(0) {
   cout << "Default Constructor called." << endl;
}
Point::Point(int x, int y) : x(x), y(y) {
   cout << "Constructor called." << endl;
}
Point::~Point() {
   cout << "Destructor called." << endl;
}
void Point::move(int newX,int newY) {
   cout << "Moving the point to (" << newX << ", " << newY << ")" << endl;
   x = newX;
   y = newY;
}
```

例6-3 对象数组应用举例

```
//6-3.cpp
#include "Point.h"
#include <iostream>
using namespace std;

int main() {
   cout << "Entering main..." << endl;
   Point a[2];
   for(int i = 0; i < 2; i++)
         a[i].move(i + 10, i + 20);
   cout << "Exiting main..." << endl;
   return 0;
}
```

运行结果：

```
Entering main...
Default Constructor called.
Default Constructor called.
Moving the point to (10, 20)
Moving the point to (11, 21)
Exiting main...
Destructor called.
Destructor called.
```

基于范围的for循环

<6.1.5>

基于范围的for循环举例

```
int main()
{
    int array[3] = {1,2,3};
    for(int & e : array)
    {
        e += 2;
        std::cout<<e<<std::endl;
    }
    return 0;
}
```

指针的概念和定义、与地址相关的运算

<6.2.1-6.2.3>

内存空间的访问方式

- 通过变量名访问
- 通过地址访问

地址运算符：&

- 例：int var;
  - &var 表示变量 var 在内存中的起始地址

指针的概念

- 指针：内存地址，用于间接访问内存单元
- 指针变量：用于存放地址的变量

指针变量

- 概念
  - **指针**：内存地址，用于间接访问内存单元
  - **指针变量**：用于存放地址的变量
- 声明和定义

  例：static int i;

  static int* ptr = &i;

  指向int变量的指针
- 引用

  例1：i = 3;

  例2：*ptr = 3;