黑猫编程 blackcat1995.com 二维数组的定义

- 当一维数组元素的类型也是一维数组时，便构成了“数组的数组”，即二维数组。二维数组定义的一般格式：

  数据类型 数组名[常量表达式1] [常量表达式2]；

  例如：int a[4][10];

  a数组实质上是一个有4行、10列的表格，表格中可储存40个元素。第1行第1列对应a数组的a[0][0]，第n行第m列对应数组元素a[n-1][m-1]。
- 说明：当定义的数组下标有多个时，我们称为多维数组，下标的个数并不局限在一个或二个，可以任意多个。
- 多维数组定义的一般格式：

  数据类型 数组名[常量表达式1] [常量表达式2] .....[常量表达式n];

  如定义一个三维数组a和四维数组b：

  int a[100][3][5];

  int b[100][100][3][5];

  多维的数组访问赋值等操作与二维数组类似。

黑猫编程 blackcat1995.com 二维数组的初始化

二维数组的初始化和一维数组类似。可以将每一行分开来写在各自的括号里，也可以把所有数据写在一个括号里。

例：

```
int  a [ 3 ]  [ 4 ] ;                    // 二维数组，3 行4列
double  b [ 2 ] [ 3 ] [ 2 ] ;            // 三维数组，2 × 3 × 2 = 12 个元素
int  i [ 2 ] [ 3 ] = { { 1, 2, 3 }, { 4, 5, 6 } } ;        // 数组初始化，推荐用
int  j [ 2 ] [ 3 ] = { 1, 2, 3, 4, 5, 6 } ;        // 与 i 数组初始化方式等价
int  k [ ] [ 2 ] [ 3 ] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 } ;    // 缺省第一维长度
int  l [ ] [ 4 ] = { { 1 }, { 1 }, { 1 } } ; // 仅对第 0 列元素赋初值
int  m [ 3 ] [ ] = { 1, 2, 3, 4, 5, 6 }     // 错误，不能省略第二维长度
```

黑猫编程 blackcat1995.com 二维数组元素的内存空间

```
const int m = 2, n = 3;
int a[m][n] = { {1, 2, 3}, {4, 5, 6} };
printf("%p %d\n", &a[0][0], a[0][0]);
printf("%p %d\n", &a[0][1], a[0][1]);
printf("%p %d\n", &a[0][2], a[0][2]);
printf("%p %d\n", &a[1][0], a[1][0]);
printf("%p %d\n", &a[1][1], a[1][1]);
printf("%p %d\n", &a[1][2], a[1][2]);
```

```
00CFF924 1
00CFF928 2
00CFF92C 3
00CFF930 4
00CFF934 5
00CFF938 6
```

1	2	3
4	5	6

1	2	3	4	5	6

黑猫编程 blackcat1995.com 二维数组的访问

- 二维数组的数组元素访问与一维数组元素访问类似，区别在于二维数组元素的访问必须给出两个下标。

  访问的格式为：

  <数组名>[下标1][下标2]

- 说明：显然，每个下标表达式取值不应超出下标所指定的范围，否则会导致致命的越界错误。

  例如,设有定义：int a[3][5];

  则表示a是二维数组（相当于一个3*5的表格），共有3*5=15个元素，它们是：

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

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

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

  因此可以看成一个矩阵（表格），a[2][3]即表示第3行第4列的元素。

快乐刷题

- P252 新矩阵
- P253 稀疏矩阵
- P254 计算矩阵边缘元素之和
- P255 矩阵转置
- P256 矩阵加法
- P257 矩阵乘法
- P258 图像旋转
- P259 图像模糊处理
- P260 图像相似度