数组是程序设计中最基础的一种数据结构。

虽然看起来简单，但在实际编程中，数组常被用来考察对底层原理的理解、内存分布的掌握、以及代码调试的能力。

什么是数组？

数组（Array）是存储在连续内存空间中的、具有相同数据类型的一组数据。

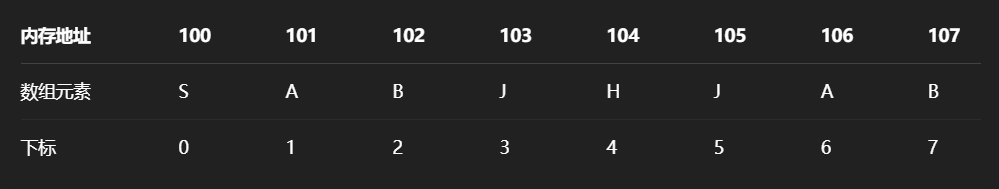
你可以把数组看成是一排整整齐齐放置的储物柜，每个柜子只能存放一种类型的数据（比如只能放 int，或者只能放 char），而且这些柜子是紧挨在一起排列的。我们可以通过“下标”来访问数组中每一个“柜子”中的数据。

示例：字符数组的内存分布

来看一个字符数组的例子：

char arr[] = {'S', 'A', 'B', 'J', 'H', 'J', 'A', 'B'};

内存中是如何存储这组字符的呢？如下图所示：

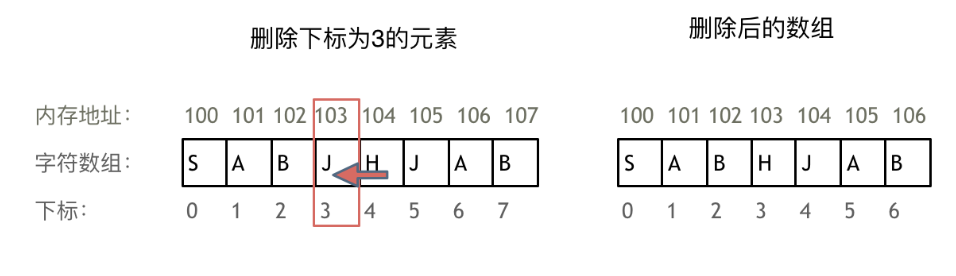


1. **数组的下标总是从 0 开始的。**
2. **数组在内存中的地址是连续的。**

因为内存地址是连续的，这意味着我们在删除数组中的某个元素，或者插入新元素时，就必须将其他元素向前或向后移动。

因为数组的内存空间是预先申请好的，大小固定。如果你说“删除”，只是逻辑意义上的——你用后面的元素把它覆盖掉，原位置的数据实际上还存在，只是不再被你使用。

例如删除下标为3的元素，需要对下标为3的元素后面的所有元素都要做移动操作，如图所示：



**array vs vector（C++ 相关）**

在 C++ 中要注意：array 和 vector 的区别很大。

* array 是固定大小的，底层就是连续的内存块；
* vector 是动态数组，虽然底层也使用连续空间，但它会在容量不足时重新分配一块更大的内存并拷贝旧内容，因此不能完全等同于 array。

**二维数组的访问方式**

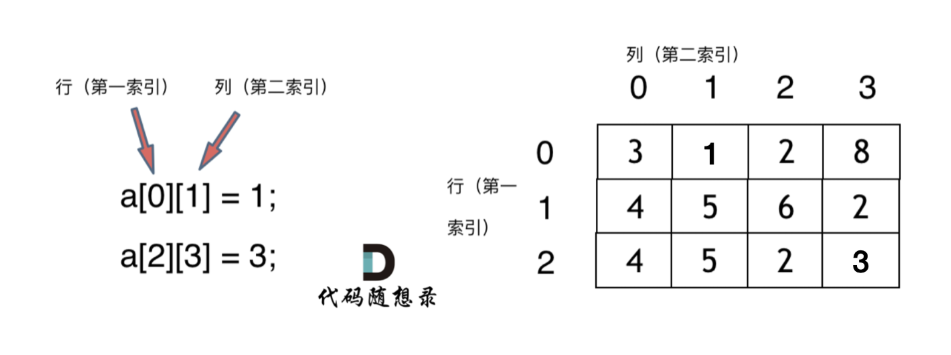
二维数组可以理解为“数组的数组”。也就是，每一行是一个一维数组，这样一整块二维数组仍然是连续的内存。

在 C++ 中，我们用两个索引访问二维数组：

a[0][1] = 1;

a[2][3] = 3;

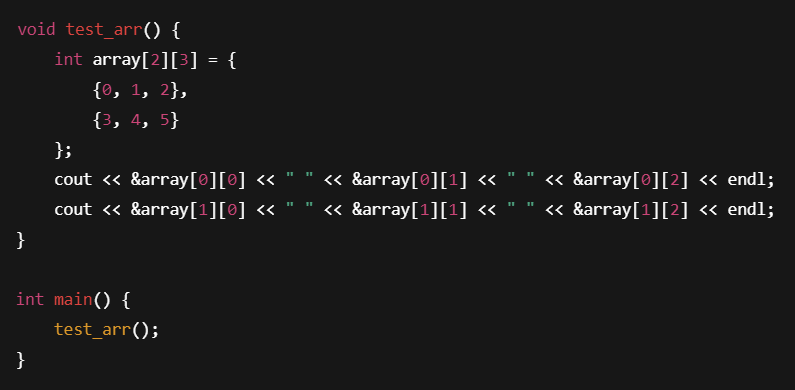
如下图所示：



**二维数组在内存中也是连续的吗？**

在 C++ 中，二维数组在内存中确实是**一行接着一行存储的**。来看一个实际的地址分布实验。

**示例代码：**



**输出的地址（示例）：**

0x7ffee4065820 0x7ffee4065824 0x7ffee4065828

0x7ffee406582c 0x7ffee4065830 0x7ffee4065834

说明地址是连续的，并且每个元素的地址相差 4 字节（因为 int 占 4 字节）。