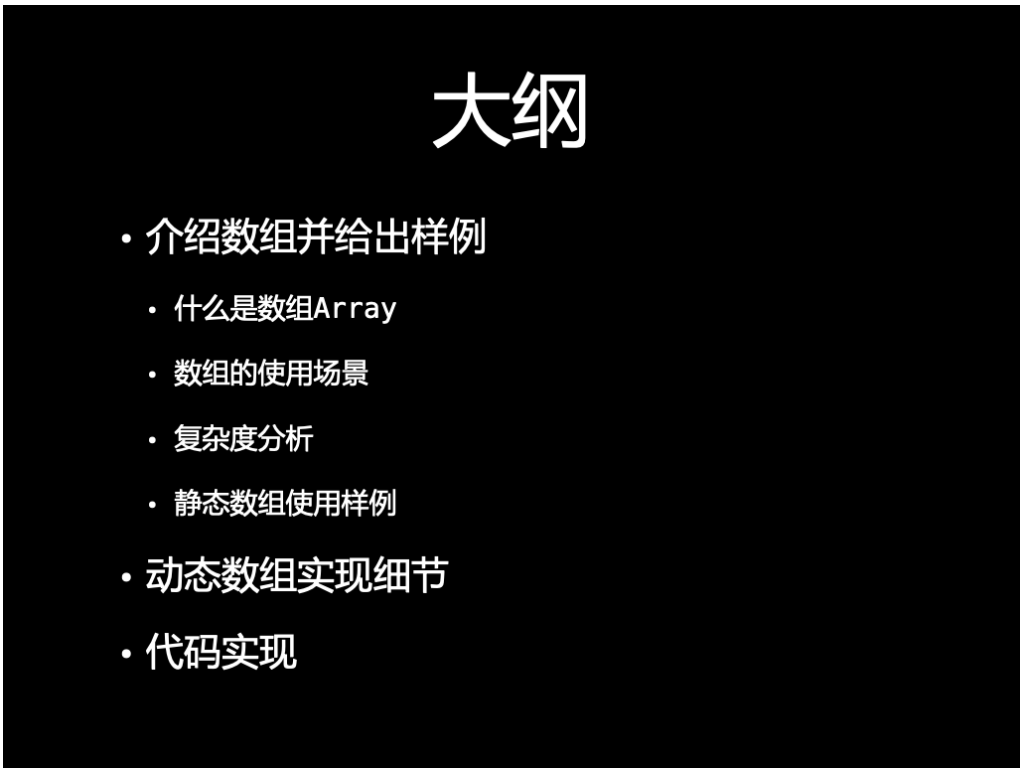

欢迎回到波波微课，今天我们来学习数组，数组可能是最常用的数据结构。本部分内容会分成两小节课，这是第一节。

数组之所以用途广泛，几乎每个程序都可以看到它的影子，主要是因为它是支持其它数据结构的基础结构。基本上只需要数组和指针，你就可以构造出其它任意的数据结构。




下面我们来看下本课的大纲。

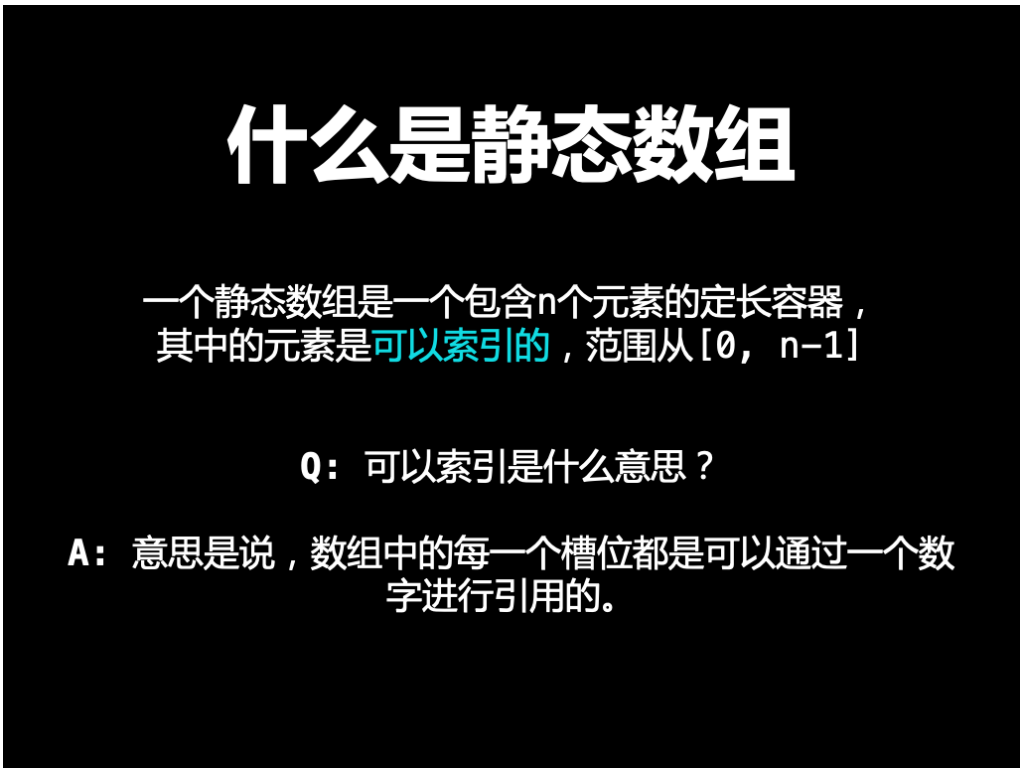
首先我会介绍什么是数组，包括数组的使用场景。

然后我会说明数组支持的主要操作，分析它们的复杂度，然后我会演示静态数组的一些使用样例。

最后，我演示动态数组是如何工作的，

然后通过代码，演示如何基于静态数组来实现动态数组。


下面我们来介绍数组，再看一些样例。

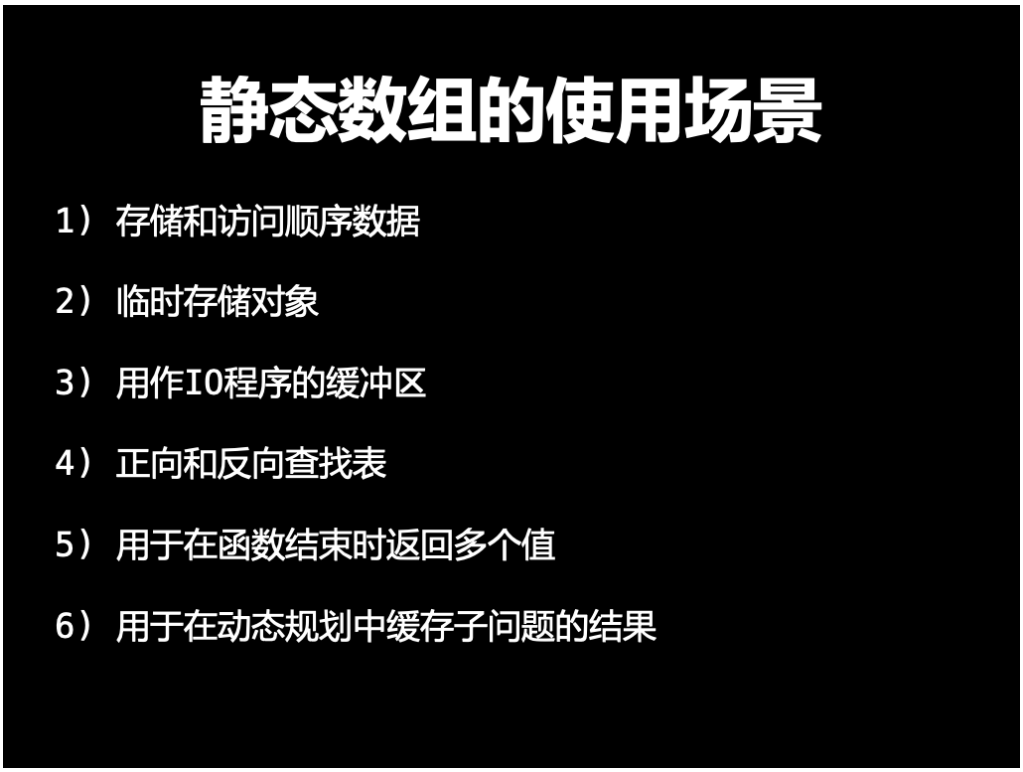



那么什么是静态数组呢？

所谓静态数组，就是一个包含n个元素的定长的容器，其中的元素是可以索引的，范围从[0, n-1]。你可能会问，什么叫可索引？其实可索引的意思是说，数组中的每一个槽位，都是可以通过一个唯一数字进行引用的。

需要进一步说明的是，静态数组的底层

实现，是采用连续的内存地址块来实现的，而不是像瑞士芝士那样有空洞和间隙。




下面来看一下静态数组的使用场景，静态数组的应用非常广泛，几乎每一个程序都可以看到它们的影子。下面是一些常见的例子。

首先，数组当然可以用于存储和访问顺序数据，其次，数组还可以存储临时对象，这些你都可能已经实际使用过了。

第三，数组也可以用作输入/输出流的缓

冲区。假设我们需要处理一个非常大的文件，这个文件太大以至于无法一下子装入内存，那么我们就可以利用数组作为缓冲buffer，每次将文件的一部分读入缓冲，进行处理，直到全部文件处理完毕。

第四，数组也可以用作查找表，因为它们具有索引特性。只需要知道数据在查找表中的索引或者偏移量，你就可以很容易从查找表找找到对应的数据。

第五，如果某种语言的函数只允许返回一个返回值，那么利用数组作为中介，通过返回数组的引用或者指针，你就可以同时返回多个返回值。

最后一个例子稍微高级一点，数组在动态规划(dynamic programming)这种高级的编程技术中，也经常被使用。它们主要用于

缓存子问题的计算结果。经典的例子是背包问题，还有零钱兑换问题。


下面我们提前来看一下静态和动态数组所支持的主要操作，并分析它们的复杂度。

对于访问操作，也就是通过索引定位元素，两个都是常量级O(1)的。

对于查找，如果不采用其它算法，直接用顺序查找，那么两个的复杂度都是O(n)，因为在最坏的情况下，我们必须

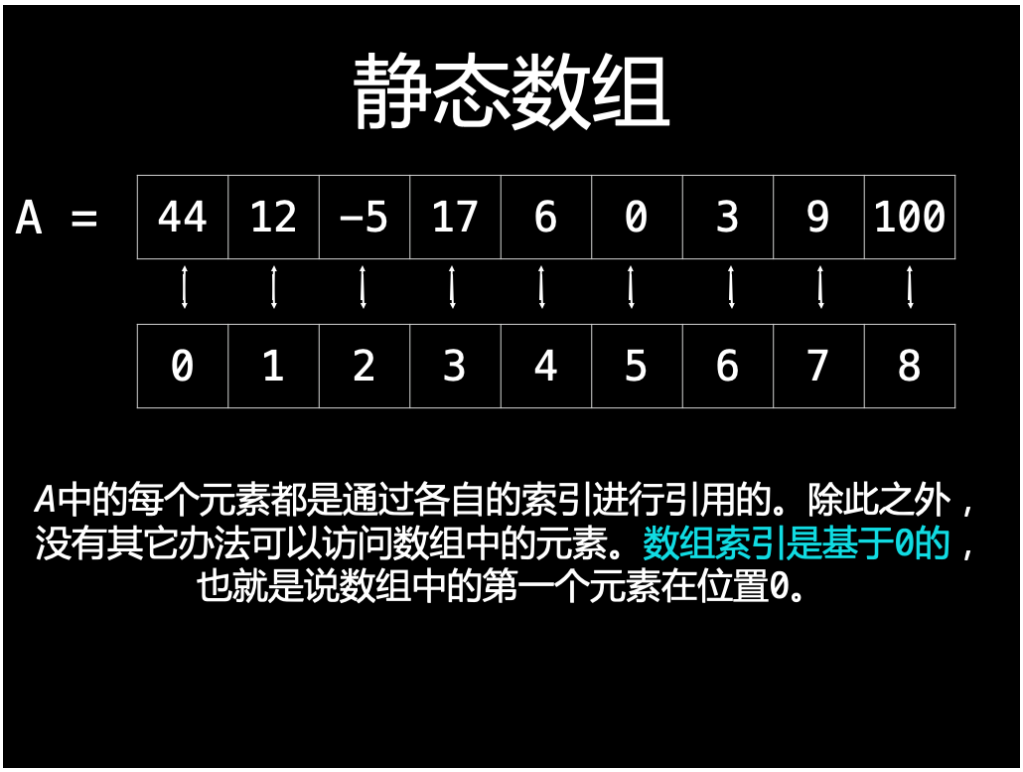
顺序查找所有元素。

对于静态数组来说，插入/添加和删除都没有意义，因为静态数组是固定的，不能变大或变小。

对于动态数组的插入Insert，它的复杂度是线性的，因为在最坏情况下，插入发生在第一个元素的位置，这时候，所有的元素都需要右移一个位置。我们这里假定动态数组是基于静态数组实现的。

对于动态数组的添加Append，它是常量级的。你可能会有疑问，因为有的时候当你添加元素的时候，内部的静态数组容量不够了，你就需要重新分配一个更大容量的静态数组。这种情况确实存在，但是它发生的情况只是少数，最后平摊下来的结果，复杂度还是常量级的。

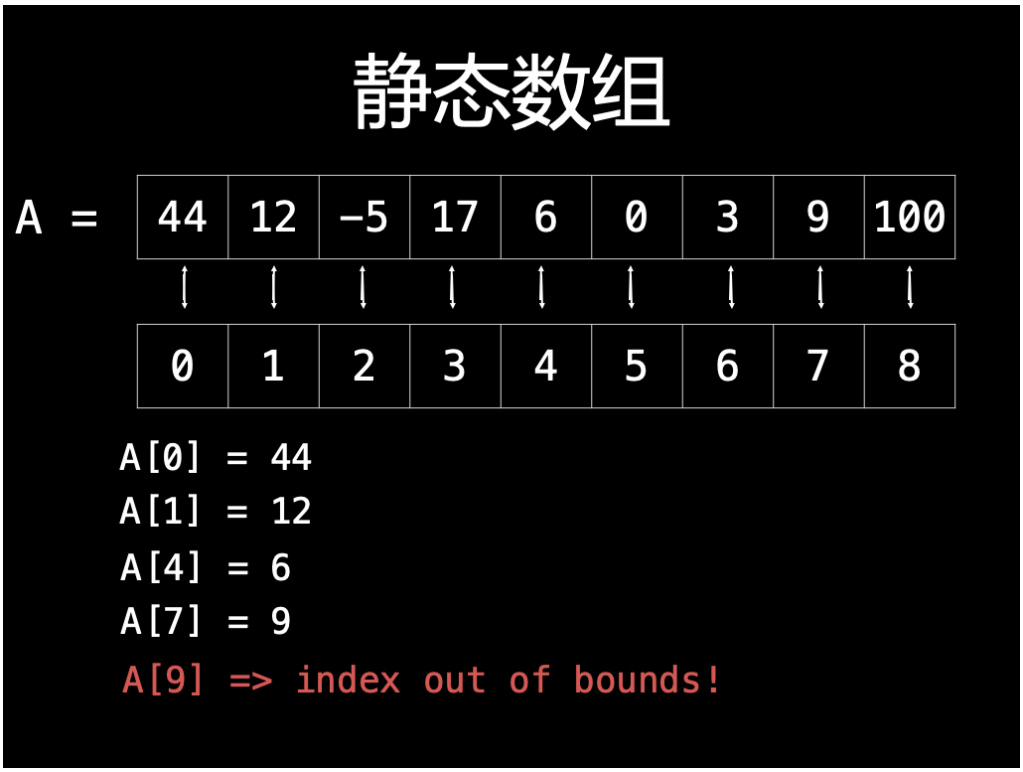
最后，和插入一样，对动态数组的删除也是线性级的，最坏的情况下，删除发生在第一个位置，这个时候所有剩下元素需要左移一个位置。




下面来看一个静态数组的样例，这个数组包含44，12，-5，17，6，0，3，9和100共9个元素。虽然这个数组的元素都是各不相同的，但是数组并没有要求元素不同。注意，第一个元素44的索引是0，而不是1。对于刚学计算机编程的同学来说，这个是最让人困惑的地方，因为通常算术都是从1开始的，但是在计算机编程语言中却是从0开始的。这个问题是由计算机科学家造成，已经变

成约定俗成的事情，大家习惯就好了。

另外在很多编程语言中，都支持使用for each循环对数组进行迭代，这时候，语句中不需要引用索引，当然实际底层还是会使用索引的，for each只是简化了语法。




对静态数组的索引访问非常简单，直接用数组名+左方括号+索引值+右方括号。

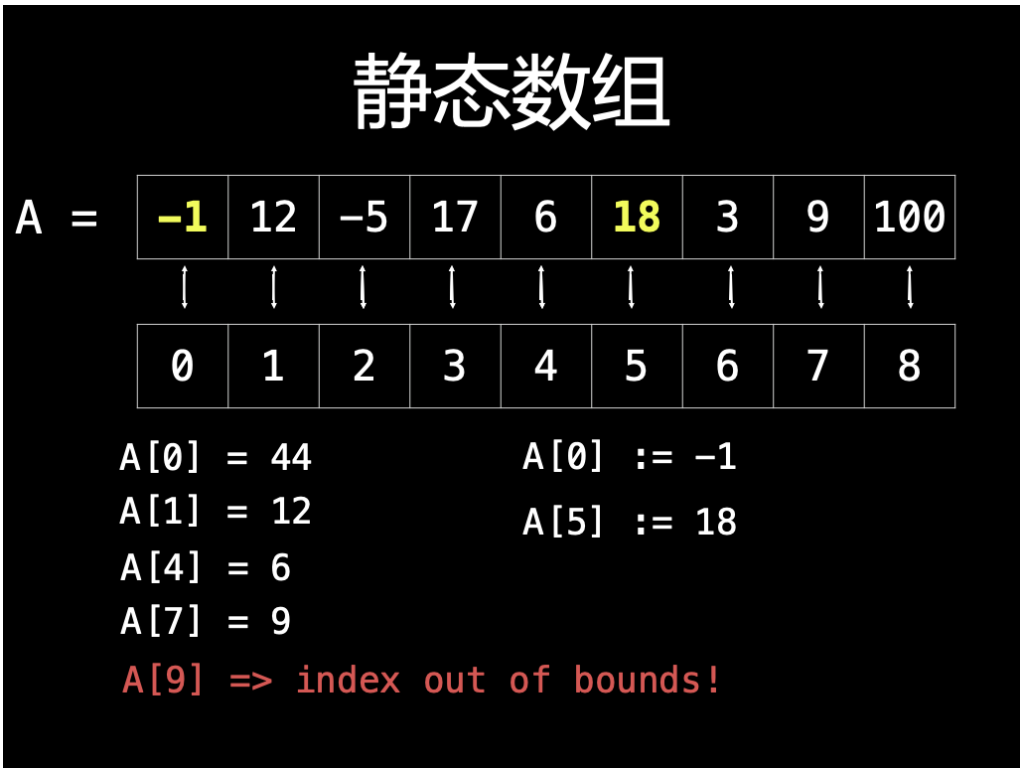
比方说：

A[0] = 44
A[1] = 12
A[4] = 6
A[7] = 9

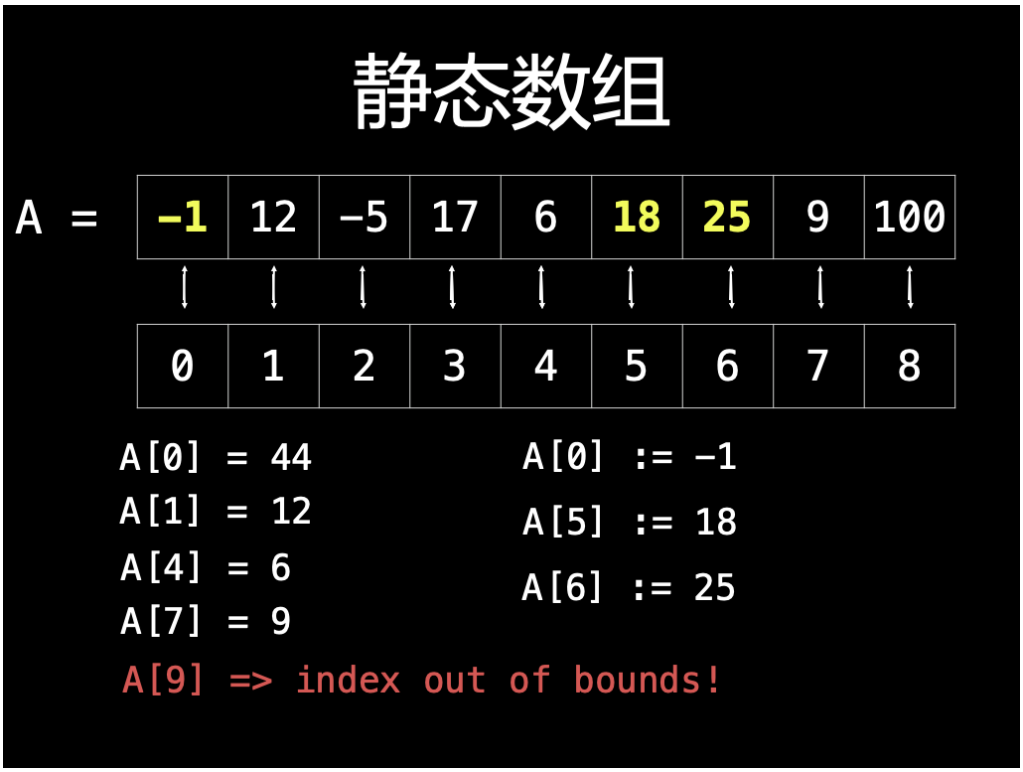
但是，当我们试图去访问A[9]的时候，会出现数组越界index out of bounds错误，因为这个索引位置不存在。有Java开发经验的同学，应该经常有机会见到ArrayIndexOutOfBoundsException。


对数组元素赋值也简单，采用类似的语法：

例如，我们把第一个元素，也就是索引为0的元素，修改为-1.

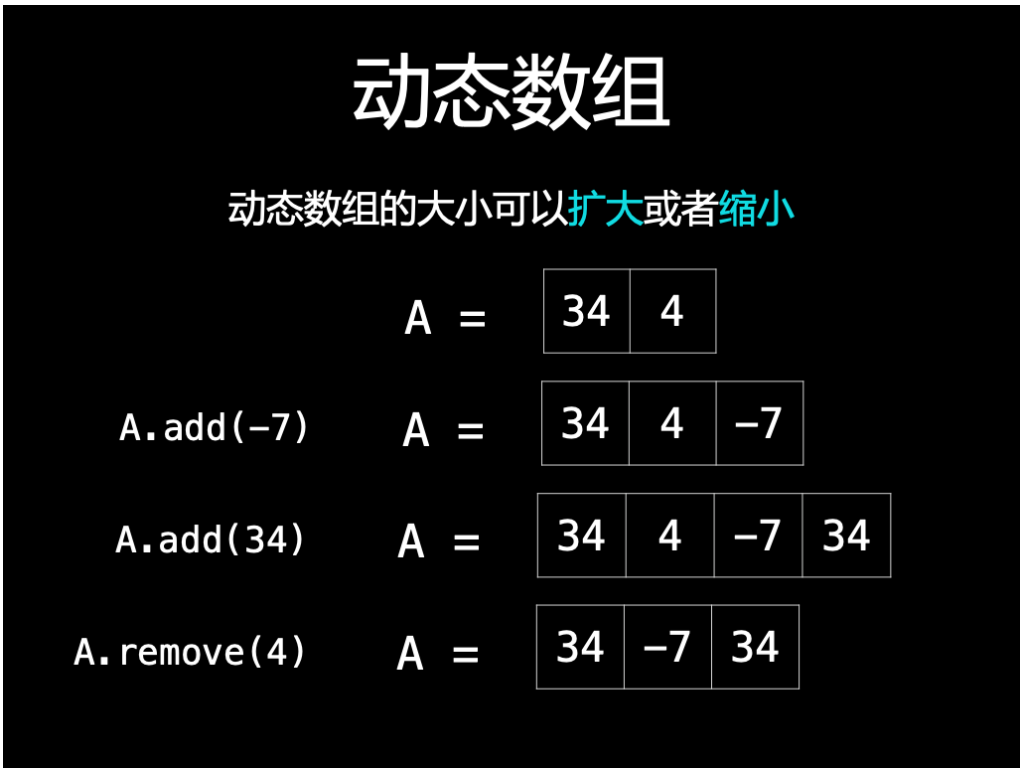



再把索引为5的元素修改为18。



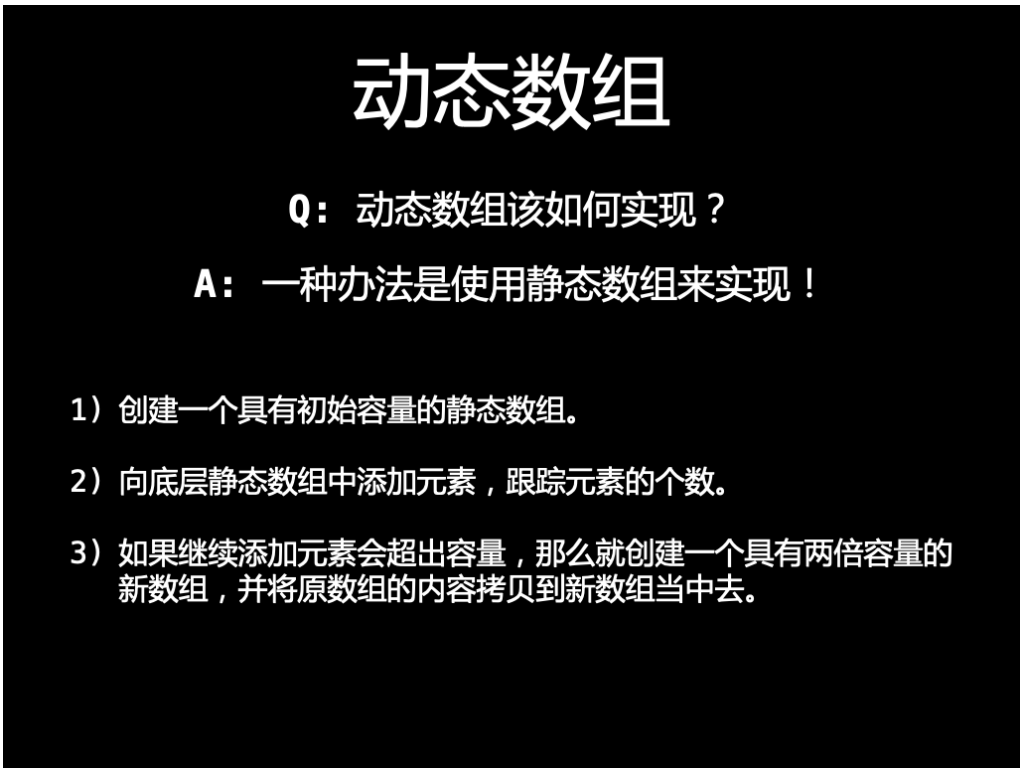

再把索引为6的元素修改为25。很简单。


下面我们来看动态数组和操作。




和静态数组一样，动态数组也可以做get和set等操作，和静态数组不同的是，动态数组的大小可以按需扩大或者缩小。

比方说PPT上的例子，刚开始A数组有两个元素，它的大小是2，然后我们在后面添加一个-7，数组的大小就增大到3个，再添加一个34，大小就增大到4个。如果移除一个4，那么数组的大小又缩小到3个。这个应该很好理解吧。




那么动态数组该如何来实现呢？一种常见的做法是使用静态数组，当然这个不是唯一的办法。

下面是使用静态数组实现的伪代码：

1. 首先先创建一个具有初始容量的静态数组，通常这个初始容量不为0。
2. 然后，我们可以按需向底层的静态数组中添加元素，并且跟踪元素的

个数

3. 后续，如果继续添加元素会超出静态数组的容量，那么就创建一个具有两倍容量的新数组，并将原数组的内容拷贝到新数组当中去。


最后，我们来看一个动态数组的演示样例，

假定我们创建了一个初始容量为2的动态数组，这个时候两个元素位置都是空的，我们用一个圈+一个反斜杠表示，然后开始向其中添加元素。。。

先添加7，占据第一个位置。再添加-9，占据第二个位置。当我们要继续添加3

的时候，静态数组的空间已经满了，所以我们再创建一个两倍大小，也就是4个元素的静态数组，先把7和-9拷贝进入，然后再把3添加到第三个位置。

然后再添加12到第四个位置。

同样，当我们要继续添加5的时候，元素位置已经占满，所以需要再创建一个大小为8的静态数组，把之前的4个元素先拷贝进去，再添加5，之后再添加-6。

在这个例子中，我们每次将静态数组的大小扩大到原来的两倍，但是其实也可以每次扩大到原来的1.5倍，或者3倍，甚至10倍。但是要注意，如果扩大的空间太大，容易造成内存空间浪费，因为有可能这些扩大的空元素位置一直用不到，结果白白占据了内存空间。

好的，这节课关于静态数组和动态数组的内容，就先讲到这边。

在下节课，波波会通过实际代码，演示如何通过静态数组实现一个动态数组类。好，我们下节课再见！