列表

本章介绍另一种线性表：列表（list）。列表和向量的区别在于，列表不要求在内存中占据的空间是连续的。这一特点赋予了列表更大的灵活性，使列表的一些操作比向量效率更高；但同时，这一特点也让列表无法通过秩定位到内存地址，丧失了循秩访问的能力，从而在另一些操作上效率不如向量。本章将详细介绍列表的特质，请读者在阅读的同时和向量进行对比，以便更好地理解列表的特点。

列表的结构

列表不要求在内存中占据的空间是连续的，因此不能循秩访问，只能循位置访问：通过指向列表中某个元素的指针（也就是该元素的地址）来访问它。那么，如何获得一个元素的地址呢？首先，把所有元素的地址汇总在一张表看起来是可行的，但如果这么做的话，这张表本身如何存储就变成了一个新的问题。如果这个表使用向量存储，那么我们就放弃了列表的不连续的灵活性；如果这个表使用列表存储，那么就成为了一个嵌套的问题。所以，我们不能把所有元素的地址汇总在一张表，也就是说，我们不能采用集中式的地址存储，需要采用分布式的地址存储。

所谓分布式的地址存储，就是我们在每个元素处，同时存储它前一个和后一个元素的地址。这样，我们无论是从前往后还是从后往前，都能遍历整个列表。很明显，这样带来了一个坏处，就是我们只能“逐个”访问元素，而不能像向量那样根据任意的秩访问元素。这就是列表选择更大灵活性的代价。除此之外，列表还有一些其他的代价，比如每个元素处除了它本身的空间，还需要存两个地址，因而有可能需要付出比向量更大的空间（注意这里是有可能，因为向量的装填因子很低时，所造成的的空间浪费更大）。为了降低列表的空间浪费，有时我们会放弃存储前一个元素的地址，只存储后一个元素的地址。这种情况称为单向列表（forward list）或单链表，相对应地，同时存储前一个和后一个元素地址的情况称为双向列表（bidirectional list）或双链表。本书中如无特别说明，列表均指双向列表。