数据结构

zcl.space

计算机编程的目的是处理数据，设计算法的目的是为了高效的处理数据。数据结构，是为了有效的组织数据。在许多场景中，我们要处理的数据可以按照一定的格式进行抽象，进而采用相同的算法。举个例子，所有的待办事项，食谱中的各种调料或者是某一门课的阅读列表，尽管这些东西类型不同，但是他们可以用相同的方式组织起来：列表。列表是一种最简单的数据结构。当然，还存在其他类型的数据结构，在计算机科学中，一些常见的数据结构包括：列表，栈，队列，集合，哈希表，树，堆，图。

我们为什么需要数据结构呢？简单而言可以归纳为三点：

1. 效率。通用的数据结构使得算法处理起来更加高效。比如，对于需要执行搜索操作的数据，一种简单的组织方法是保存在列表中，在搜索时，从头至尾，逐个遍历。但是这种方法是非常低效的方法，尤其对于大量的数据而言。如果我们把这些数据以其他数据结构保存，比如哈希表或者二叉树，我们可以用更快的算法搜索。

2. 抽象。数据结构提供了一种更易理解的方式保存数据。在解决问题时，数据结构在一定程度上对数据进行了抽象提取。比如，用栈保存数据，我们可以关注针对栈的操作（压栈和出栈），而不是这些操作是如何实现的。数据结构可以让我们在更高的抽象层次处理数据。

3. 复用。数据结构的实现是模块化的，因此是可以复用的。对于每一种数据结构，能够执行的操作是预先定义好的，也就是说，在实用数据结构时，只能实用预先定义好的操作。另外数据结构也是上下文无关的。因为他们可以被用于任何场景的任何数据。在C语言中，我们实用指针来指向数据而不是维护同一份数据的多份拷贝。

当提起某种数据结构时，我们应该了解这种数据结构支持的特定操作，然后我们才能更好的决定是不是要采用这种数据结构。比如，对于列表，我们很自然的会想到：插入列表，从列表中移除，便利列表和统计列表元素。所以，对于带有这些基本操作的数据结构，我们又称为抽象数据类型（abstract datatype，ADT）。ADT中支持的这些操作叫做公共接口。这些ADT的公共接口定义了我们可以用这种数据结构完成的操作。构建数据类型的接口是非常重要的，因为这些接口决定了后续代码的可读性和可维护性。