**数组和链表区别**

**数组静态分配内存，链表动态分配内存；**

**数组在内存中连续，链表不连续；**

**数组元素在栈区，链表元素在堆区；**

**数组利用下标定位，时间复杂度为O(1)，链表定位元素时间复杂度O(n)；**

**数组插入或删除元素的时间复杂度O(n)，链表的时间复杂度O(1)。**

谈到链表与数组的区别，可以从几个不同的角度来谈，

首先从逻辑结构上说，两者都是数据结构的一种，但存在区别，

数组是申请的一块连续的内存空间，并且是在编译阶段就要确定空间大小的，同时在运行阶段是不允许改变的，所以它不能够随着需要的改变而增加或减少空间大小，所以当数据量大的时候，有可能超出了已申请好的数组上限，产生数据越界，或者是数据量很小，对于没有使用的数组空间，造成内存浪费。

链表则是动态申请的内存空间，并不像数组一样需要事先申请好大小，链表是现用现申请就OK，根据需求动态的申请或删除内存空间，对于的是增加或删除数据，所以比数组要灵活。

再从物理存储即内存分配上分析，

数组是连续的内存，对于访问数据，可以通过下标直接读取，时间复杂度为O(1)，而添加删除数据就比较麻烦，需要移动操作数所在位置后的所有数据，时间复杂度为O（N）。

链表是物理上非连续的内存空间，对于访问数据，需要从头便利整个链表直到找到要访问的数据，没有数组有效，但是在添加和删除数据方面，只需要知道操作位置的指针，很方便可以实现增删，教数组比较灵活有效率。

所以综合以上，对于快速访问数据，不经常有添加删除操作的时候选择数组实现，而对于经常添加删除数据，对于访问没有很高要求的时候选择链表。

**数组和链表的区别**

数组是将元素在内存中连续存放，由于每个元素占用内存相同，可以通过下标迅速访问数组中任何元素。但是如果要在数组中增加一个元素，需要移动大量元素，在内存中空出一个元素的空间，然后将要增加的元素放在其中。同样的道理，如果想删除一个元素，同样需要移动大量元素去填掉被移动的元素。如果应用需要快速访问数据，很少或不插入和删除元素，就应该用数组。

链表恰好相反，链表中的元素在内存中不是顺序存储的，而是通过存在元素中的指针联系到一起。比如：上一个元素有个指针指到下一个元素，以此类推，直到最后一个元素。如果要访问链表中一个元素，需要从第一个元素开始，一直找到需要的元素位置。但是增加和删除一个元素对于链表数据结构就非常简单了，只要修改元素中的指针就可以了。如果应用需要经常插入和删除元素你就需要用链表数据结构了。

   \*C++语言中可以用数组处理一组数据类型相同的数据，但不允许动态定义数组的大小，即在使用数组之前必须确定数组的大小。而在实际应用中，用户使用数组之前有时无法准确确定数组的大小，只能将数组定义成足够大小，这样数组中有些空间可能不被使用，从而造成内存空间的浪费。链表是一种常见的数据组织形式，它采用动态分配内存的形式实现。需要时可以用new分配内存空间，不需要时用delete将已分配的空间释放，不会造成内存空间的浪费。

　　(1) 从逻辑结构角度来看

　　   a, 数组必须事先定义固定的长度（元素个数），不能适应数据动态地增减的情况。当数据增加时，可能超出原先定义的元素个数；当数据减少时，造成内存浪费。

　　   b,链表动态地进行存储分配，可以适应数据动态地增减的情况，且可以方便地插入、删除数据项。（数组中插入、删除数据项时，需要移动其它数据项）

　　(2)从内存存储角度来看

　　   a,(静态)数组从栈中分配空间, 对于程序员方便快速,但自由度小。

　　   b, 链表从堆中分配空间, 自由度大但申请管理比较麻烦.

链表的效率为何比数组低些？先从两者的初始化开始。数组无需初始化，因为数组的元素在内存的栈区，系统自动申请空间。而链表的结点元素在内存的堆区，每个元素须手动申请空间，如malloc。也就是说数组是静态分配内存，而链表是动态分配内存。链表如此麻烦为何还要用链表呢？数组不能完全代替链表吗？回到这个问题只需想想我们当初是怎么完成学生信息管理系统的。为何那时候要用链表？因为学生管理系统中的插入，删除等操作都很灵活，而数组则大小固定，也无法灵活高效的插入，删除。因为堆操作灵活性更强。数组每次插入一个元素就需要移动已有元素，而链表元素在堆上，无需这么麻烦。

说了这么多，数组和链表的区别整理如下：

**数组静态分配内存，链表动态分配内存；**

**数组在内存中连续，链表不连续；**

**数组元素在栈区，链表元素在堆区；**

**数组利用下标定位，时间复杂度为O(1)，链表定位元素时间复杂度O(n)；**

**数组插入或删除元素的时间复杂度O(n)，链表的时间复杂度O(1)。**