### 1、什么是链表

链表（Linked list）是一种线性表，但是并不会按线性的顺序存储数据，而是在每一个节点里存到下一个节点的指针(Pointer)。

插入复杂度O(1)；查找一个节点或者访问特定编号的节点复杂度O(n)。

与数组比较：

* 从底层的存储结构上来看：数组需要一块儿连续的内存空间，堆内存的要求比较高；而链表恰恰相反，它并不需要一块儿连续的内存空间，它通过“指针”将一组零散的内存块串联起来。
* 使用链表结构可以克服数组链表需要预先知道数据大小的缺点，链表结构可以充分利用计算机内存空间，实现灵活的内存动态管理。但是链表失去了数组随机读取的优点，同时链表由于增加了结点的指针域，空间开销比较大。

常见的链表结构：

* 单链表
* 单向循环链表
* 双向循环链表

链表应用场景：适合存储对元素查找，访问要求低，但对删除，插入要求高的数据。

一、单链表

一个简单的结点结构如图所示，是构成单链表的基本结点结构。



数据域：存储数据元素；

指针域：指向下一个具有相同结构的结点。



* 头结点：有时，在链表的第一个结点之前会额外增设一个结点，结点的数据域一般不存放数据（有些情况下也可以存放链表的长度等信息）。
* 头指针：永远指向链表中第一个结点的位置（如果链表有头结点，头指针指向头结点；否则，头指针指向首元结点）。
* 首元结点：链表中第一个元素所在的结点，它是头结点后边的第一个结点。

若头结点的指针域为NULL，表明链表时空表。头结点对于链表来说，不是必须

的，在处理某些问题时，给链表添加头结点会使问题变得简单。

头指针只声明而没有分配存储空间，头结点进行了声明并分配了一个结点的实际物理内存。

头结点的引入能使链表对第一个元素的删除和插入和其他元素相同，不用另外说明，使得代码更加简洁。

### 1、基本操作

#### 1.1 单链表增添元素

1. 声明一个新节点node作为新的尾结点，next=null；
2. 获取原链表的最后一个节点，把它的next指向1步骤的新结点node；
3. 记录链表长度的变量+1。

#### 1.2 单链表插入元素

1. 获取需要插入的位置的节点；
2. 声明一个新节点node指向1步骤得到的节点；
3. 获取需要插入位置节点的上一个节点；
4. 将3步骤得到的节点的next指向新节点node；
5. 记录链表长度的变量+1。

public class MySingleLinkList<T> {

*/\*\*  
 \* 首元结点  
 \*/* private Node<T> first;  
  
 */\*\*  
 \* 头指针  
 \*/* private Node<T> head;  
  
 */\*\*  
 \* 链表长度  
 \*/* int size;  
  
 private final static int *POINTER\_INDEX* = -2;  
  
 public boolean initList() {  
 size = 0;  
 first = new Node<>(null, null);  
 head = new Node<>(null, first);  
 return true;  
 }  
  
 public boolean isEmpty() {  
 return size == 0;  
 }  
  
 */\*\*  
 \* 获取结点  
 \*/* public Node<T> getNode(int index) {  
 Node<T> renode = head;  
 // 移动指针位置，初始位置为-2  
 for (int p = *POINTER\_INDEX*; p < index; p++) {  
 renode = renode.next;  
 }  
 return renode;  
 }  
  
 */\*\*  
 \* 在末尾添加元素  
 \*/* public void add(T a) {  
 // 1.声明一个新结点作为新的尾结点，next = null  
 Node<T> renode = new Node<>(a, null);  
 // 2.获取原链表的最后一个结点，把它的next指向新结点  
 getNode(size - 1).next = renode;  
 // 3.长度+1  
 size++;  
 }  
  
 */\*\*  
 \* 删除i位置结点，并返回删掉的数据  
 \*/* public T remove(int index) {  
 // 如果是尾结点，将前一个结点的指针域设置为null，返回尾结点的data域  
 if (index == size - 1) {  
 T delData = getNode(size - 1).data;  
 getNode(size - 2).next = null;  
 return delData;  
 }  
  
 Node<T> prev = getNode(index - 1);  
 T delData = prev.next.data;  
 prev.next = prev.next.next;  
 size--;  
 return delData;  
 }

}

二、链表面试题

### 1、链表逆序

### 2、链表复制

### 3、链表合并

### 4、链表相交判断

### 5、判断链表是否存在环

### 6、链表模拟加法

### 7、有序单向循环链表插入结点

### 8、输出链表倒数第k个结点

三、参考资料

<https://www.geeksforgeeks.org/detect-loop-in-a-linked-list/>

<http://www.cppblog.com/humanchao/archive/2008/04/17/47357.html>

<https://www.geeksforgeeks.org/find-first-node-of-loop-in-a-linked-list/>

<https://www.geeksforgeeks.org/merge-two-sorted-linked-list-without-duplicates/>