链表基础

什么是链表?

链表是一种通过指针串联在一起的线性结构，每一个节点由两部分组成，一个是数据域，一个是指针域（用于存放指向下一个节点的指针），最后一个节点的指针域指向nullptr（空指针）。

链接的入口节点称为链表的头结点也就是head。

一般来说，还要区分以下概念：

- 头结点（head）：不携带任何实际数据，只用于工具使用
- 首元结点：第一个携带实际数据的结点

链表分类

链表分为好多种类。

单链表

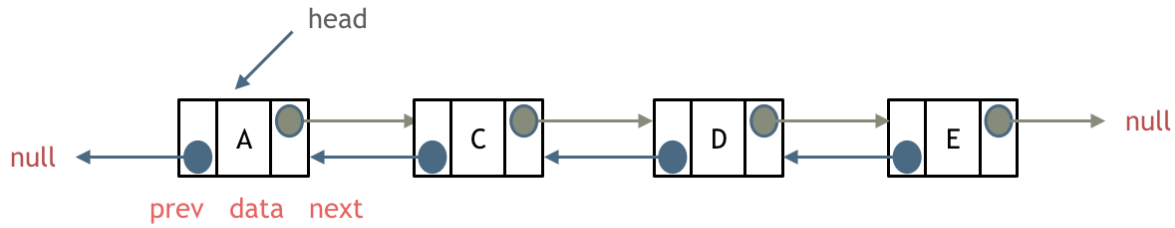
最基本的链表 前面已经提到过了

双链表

单链表中的节点只能指向节点的下一个节点。

而双链表中的每一个节点由两个指针域，一个指向下一个节点，另一个指向上一个节点。

双链表既可以向前查询也可以向后查询。

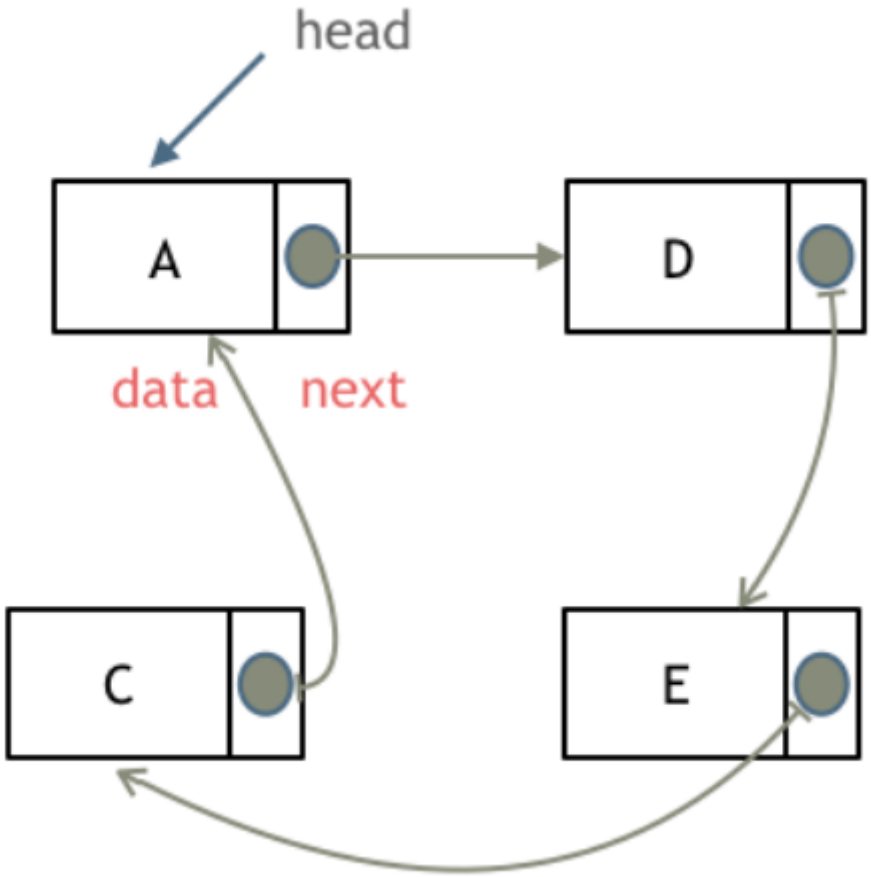


prev--前驱 next--后继

循环链表

循环链表，顾名思义，就是链表首尾相连。

循环链表可以用于解决约瑟夫环问题



链表的存储方式

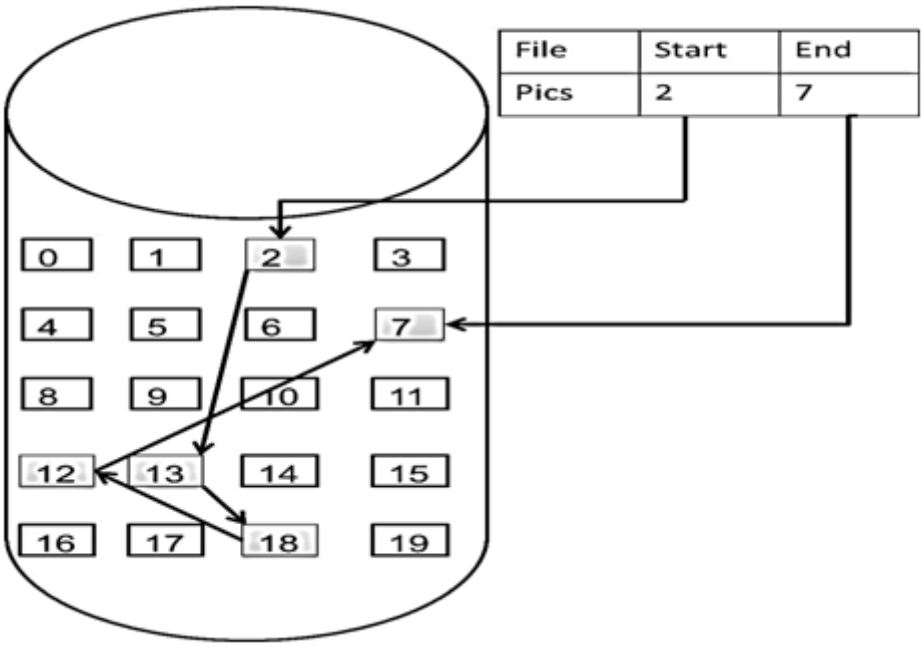
让我们来看看链表在内存中的存储方式。

数组是在内存中是连续分布的，但是链表在内存中可不是连续分布的

链表是线性表的链接存储结构的实现

链表是通过指针域的指针链接在内存中各个节点

所以链表中的节点在内存中不是连续分布的，而是散乱分布在内存中的某地址上，分配机制取决于操作系统的内存管理。



链表的定义

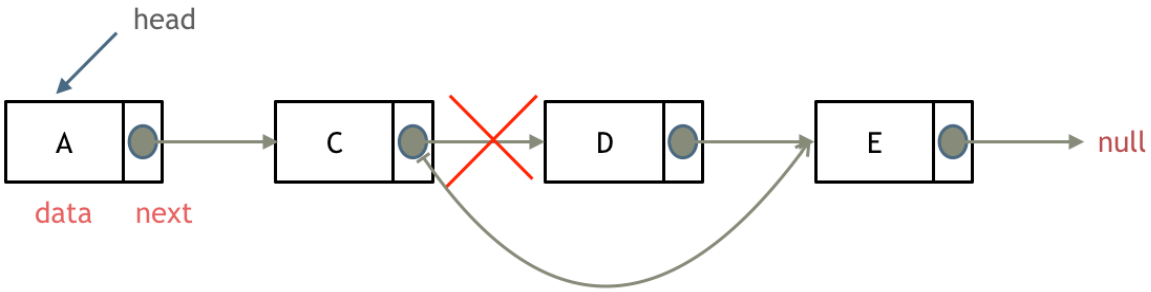
来看一种标准的C++式链表定义

```
// 单链表
struct ListNode {
    int val;  // 节点上存储的元素
    ListNode *next;  // 指向下一个节点的指针
    ListNode(int x) : val(x), next(NULL) {}  // 节点的构造函数
};
```

链表的操作

删除节点

删除D节点，如图所示：



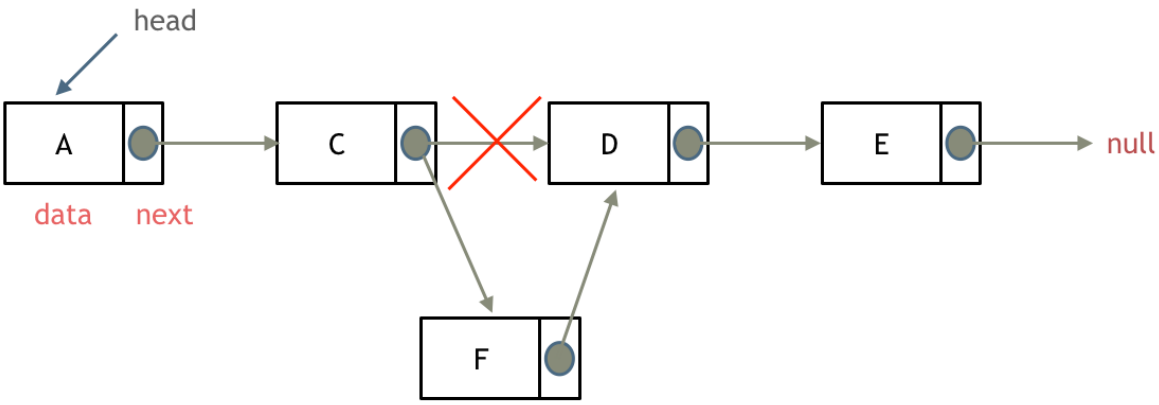
只要将C节点的next指针 指向E节点就可以了。

可是D节点不是依然存留在内存里么？只不过是没有在这个链表里而已。

是这样的，所以在C++里最好是再手动释放这个D节点，释放这块内存。

添加节点

如图所示：



可以看出链表的增添和删除都是$O(1)$操作，也不会影响到其他节点。

但是要注意，要是删除第五个节点，需要从头节点查找到第四个节点通过next指针进行删除操作，查找的时间复杂度是$O(n)$。

性能分析

再把链表的特性和数组的特性进行一个对比，如图所示：

	插入/删除（时间复杂度）	查询（时间复杂度）	适用场景
数组	O(n)	O(1)	数据量固定，频繁查询，较少增删
链表	O(1)	O(n)	数据量不固定，频繁增删，较少查询

数组在定义的时候，长度就是固定的，如果想改动数组的长度，就需要重新定义一个新的数组。

链表的长度可以是不固定的，并且可以动态增删， 适合数据量不固定，频繁增删，较少查询的场景。