

顺序表:

↳ 相当于数组

结构定义: { 1. size → 总长度
2. length → 当前长度
3. data-type 人为规定

结构操作: { 1. 插入
2. 删除

✗ 结构操作实现之前先进行合法性判断

✗ 插入操作时, 不能以正序元素后移的方式插入, 这种情况会将当前元素的后一位元素覆盖掉, 应以后序将元素后移 先试着在后面申请空间

realloc →

申请失败，调用 malloc 申请空间
数据拷贝，释放 p



依旧申请失败，返回 null，不动原空间

链表：

内存内部：链表真正存储信息结点
程序内部：索引内存内部链表结构的小抓手

内存内部开辟几个结点，结点之间有指向关系

结构定义 { data-type
指针域 (存储下一个结点的地址)

结构操作

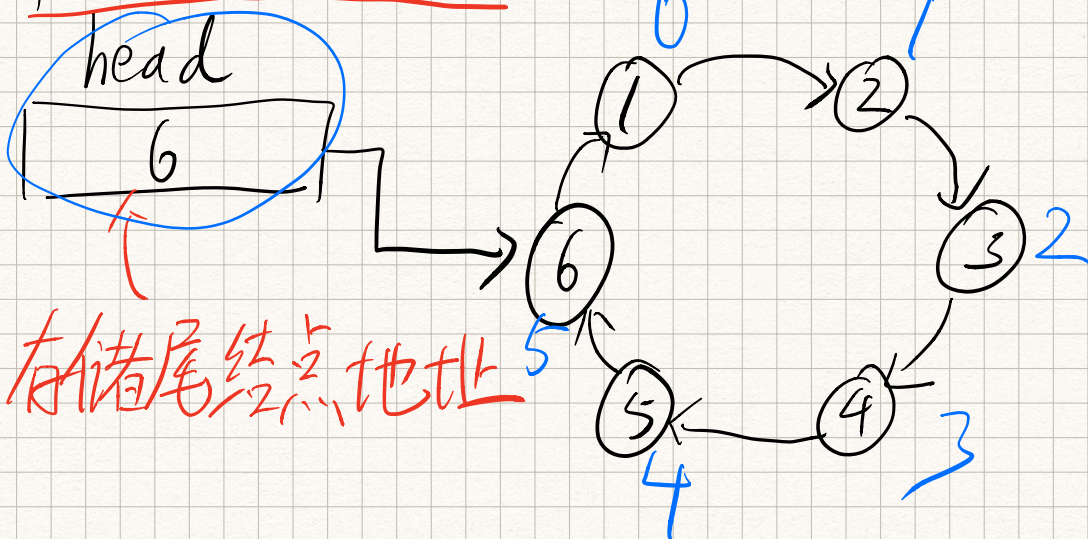
insert

(先操作待插入结点，
再操作原位置结点，
顺序反了则会丢失后序
结点信息，造成内存泄露)

erase

(待删除结点的前一个
结点指向待删除结点的
下一个结点)

单向循环链表



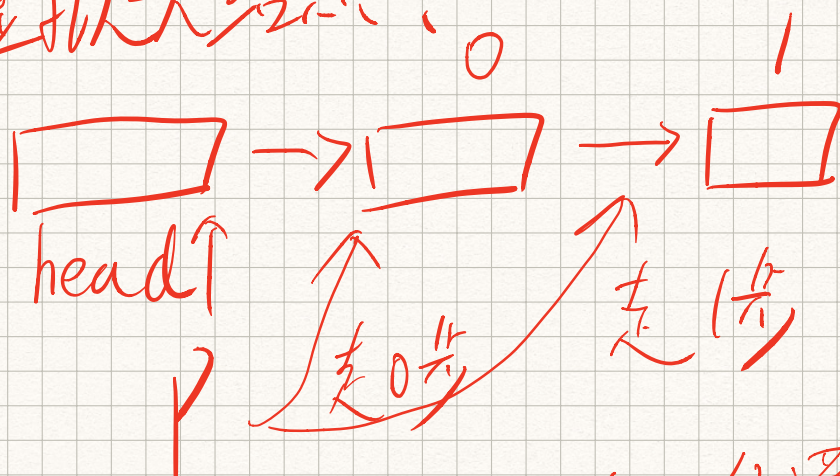
有诸尾结点地址

13 0 2 1 2 4 1 → 5 12 10

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32

(需要遍历一圈, 不合理)

✖ 虚拟头结点: 0



指针1会站在待插入位置的前一个位置