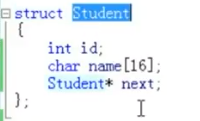
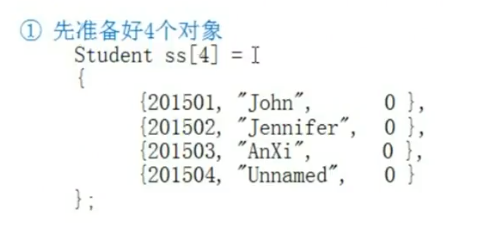
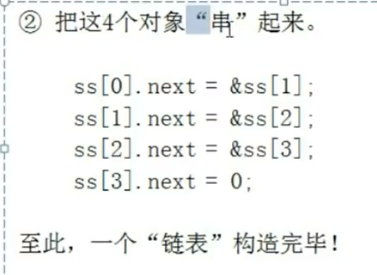
删除节点的就方法就是 挂在在前一个节点上 然后直接free删除当前节点

尽量使用有头链表 无头节点的删除和插入较为麻烦

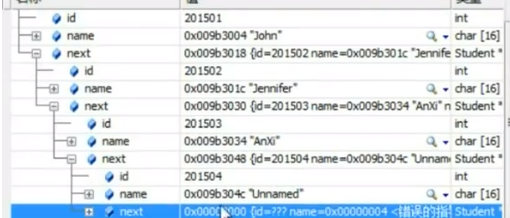
//先声明好 结构

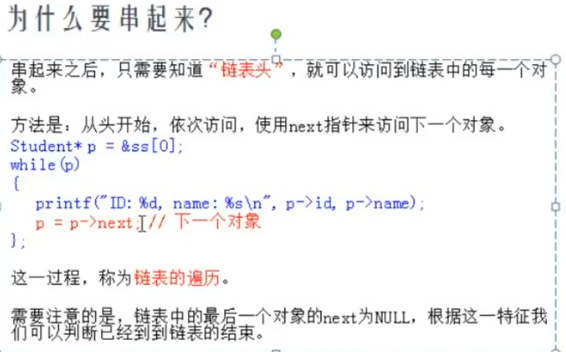


//创建好 结构对象的数据



//链表的构造



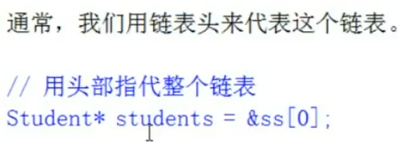


当 列表构造好后可以使用 首个 地址去遍历

链表的特征

链表头：指链表中的第一个对象

链表尾 指链表中的最后一个对象。它的next必须设为空指针NULL



查找链表的某个对象语法

Student \* find (Student\* head, int id) // 链表头 ，要查找的结构数据成员类型

{

Student \* p = head;

while(p)

{

if (p ->id ==id) //符合条件

return p;

p->next = p;/\* code \*/

}

return NULL; //没找到符合条件的对象

}

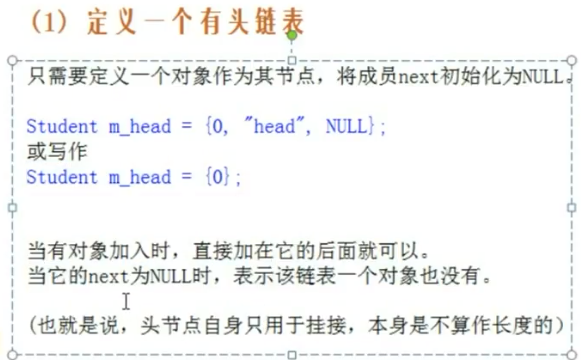
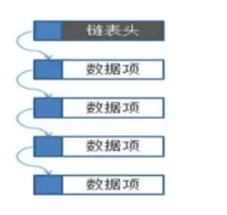


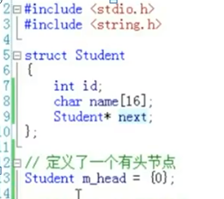
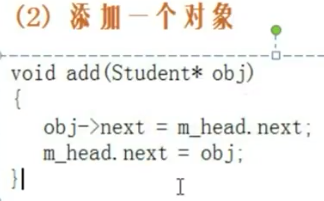
无头链表 和 有头链表

无头链表表示

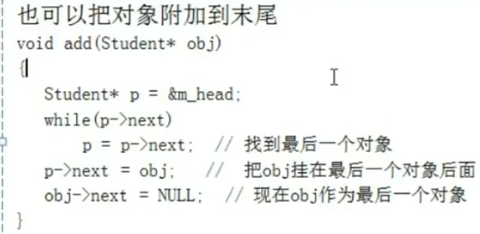
如上面定义的 环环相扣

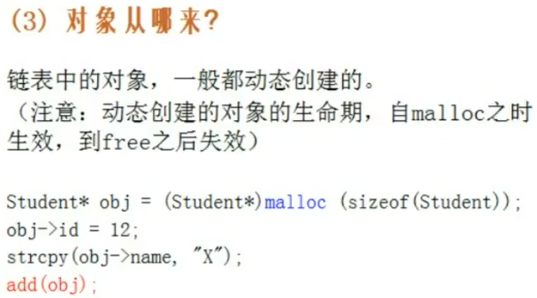
有头链表 //第一个没有数据的只是作为地址

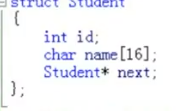


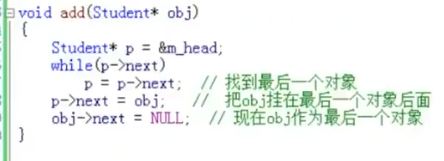
 

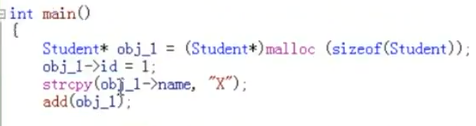
如果过要把新创建的对象放在链表的最后一个 用遍历方式





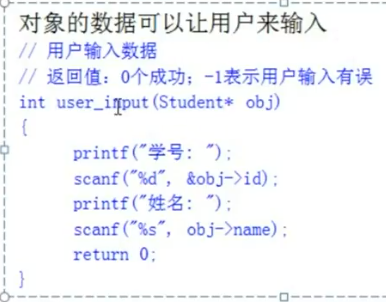




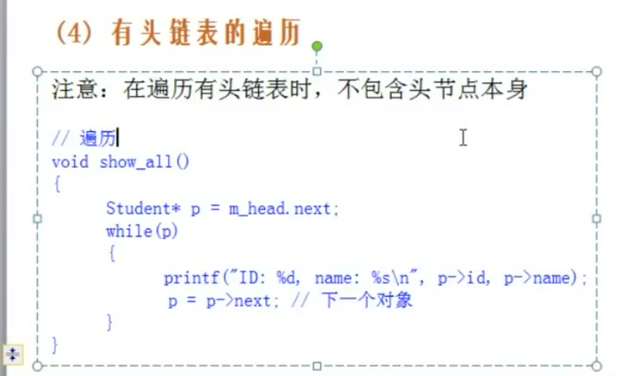


前面的是 定义好 数据用ctrcpy

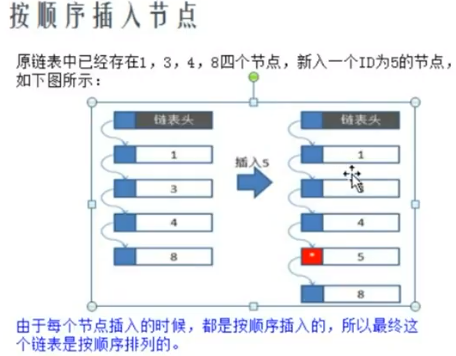
定义一个函数 user\_input用scanf 用户输入来定义 数据





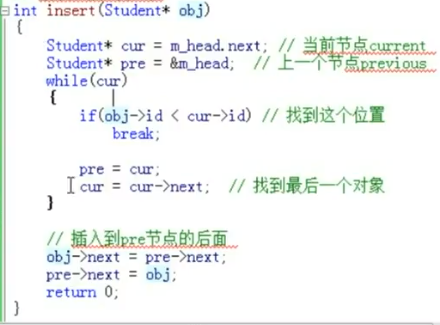


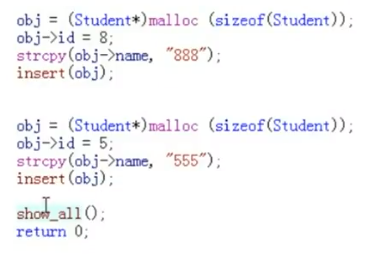
插入节点



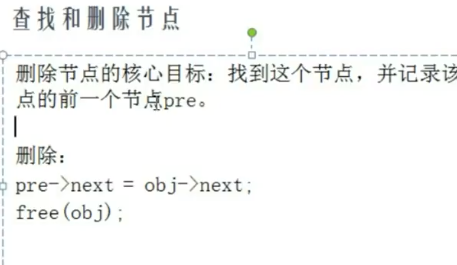
思维和值交换是一样的 节点跟节点 交换。

目的是// 当没有按顺序创建 对象的结构数据 使用顺序节点也可以按照指定的顺序输出排序

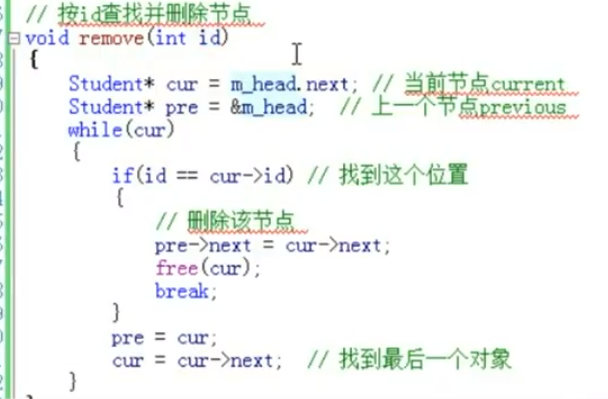


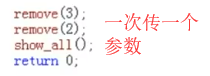
//原型

查找删除节点



把现在节点obj的内容挂在 上一个节点 然后 删除现在的节点





对比数组的优势

