逻辑教育
Logic education

Hello 数据结构与算法

数据结构与算法 — 线性表[专题] 数据结构与算法主题[2]

@HelloCoder_CC

全力以赴，非同凡"想"

课程研发:CC老师
课程授课:CC老师

线性表—顺序表[01]

对于非空的线性表和线性结构,其特点如下:

- 存在唯一的一个被称作"第一个"的数据元素;
- 存在唯一的一个被称作"最后一个"的数据元素
- 除了第一个之外,结构中的每个数据元素均有一个前驱
- 除了最后一个之外,结构中的每个数据元素都有一个后继.

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

ADT List {

Data: 线性表的数据对象集合为*{a1,a2,.....an}*,每个元素的类型均为*DataType.* 其中,除了第一个元素*a1*外,每一个元素有且只有一个直接前驱元素,除了最后一个元素*an*外,每个元素有且只有一个直接后继元素. 数据元素之间的关系是一对一的关系.

Operation

InitList(&L)

操作结果:初始化操作,建立一个空的线性表*L.*

DestroyList(&L)

初始条件: 线性表*L*已存在

操作结果: 销毁线性表*L.*

ClearList(&L)

初始条件: 线性表*L*已存在

操作结果: 将*L*重置为空表.

ListEmpty(L)

初始条件: 线性表*L*已存在

操作结果: 若*L*为空表,则返回*true*,否则返回*false.*

ListLength(L)

初始条件: 线性表*L*已存在

操作结果: 返回*L*中数据元素的个数

……

……

GetElem(L,i,&e)
初始条件: 线性表*L*已存在,且*1<=i<ListLength(L)*
操作结果: 用*e*返回*L*中第*i*个数据元素的值;

LocateElem(L,e)
初始条件: 线性表*L*已存在
操作结果: 返回*L*中第*1*个值与*e*相同的元素在*L*中的位置. 若数据不存在则返回*0*;

PriorElem(L, cur_e,&pre_e);
初始条件: 线性表*L*已存在
操作结果: 若*cur_e*是*L*的数据元素,且不是第一个,则用*pre_e*返回其前驱,否则操作失败.

NextElem(L, cur_e,&next_e);
初始条件: 线性表*L*已存在
操作结果: 若*cur_e*是*L*的数据元素,且不是最后一个,则用*next_e*返回其后继,否则操作失败.

……

课程研发:CC老师
课程授课:CC老师

.....

ListInsert(L,i,e);
初始条件*:* 线性表*L*已存在,且*1<=i<=listLength(L)*
操作结果*:* 在*L*中第*i*个位置之前插入新的数据元素*e*,*L*长度加*1.*

ListDelete(L,i);
初始条件*:* 线性表*L*已存在,且*1<=i<=listLength(L)*
操作结果*:* 删除*L*的第*i*个元素,*L*的长度减*1.*

TraverseList(L);
初始条件*:* 线性表*L*已存在
操作结果*:* 对线性表*L*进行遍历,在遍历的过程中对*L*的每个结点访问*1*次*.*

}ADT List.

逻辑教育
Logic education

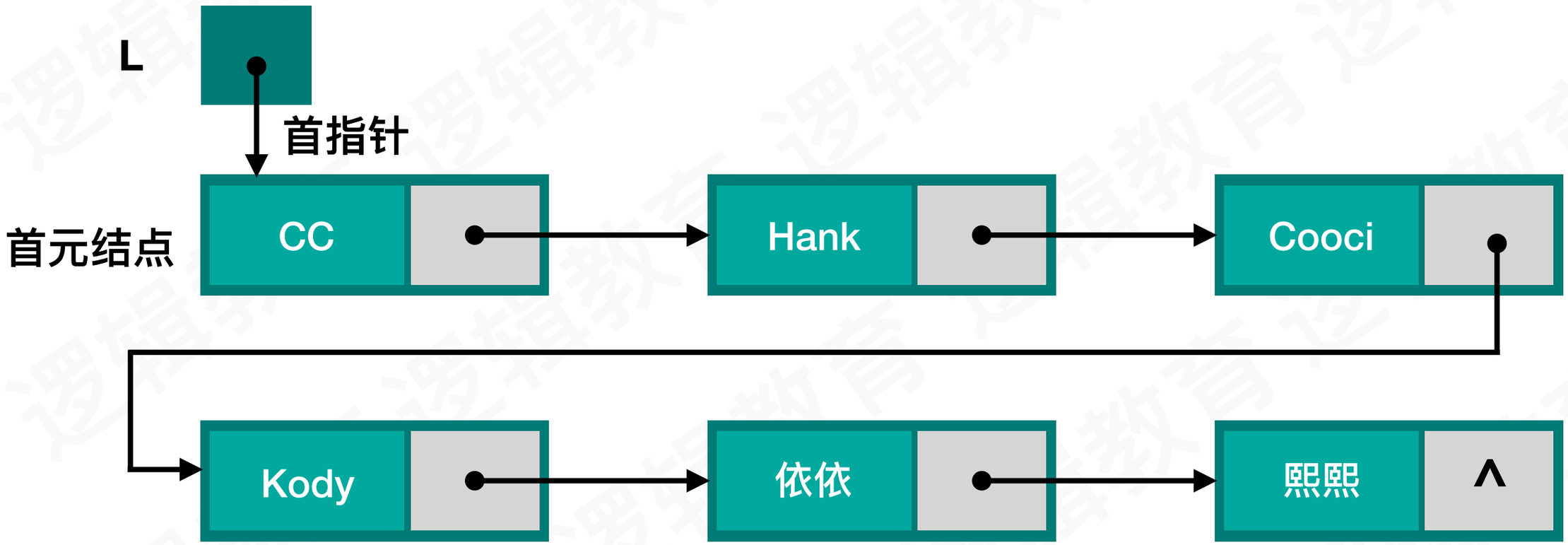
线性表—单链表结点

结点

数据域　　指针域

逻辑教育
Logic education

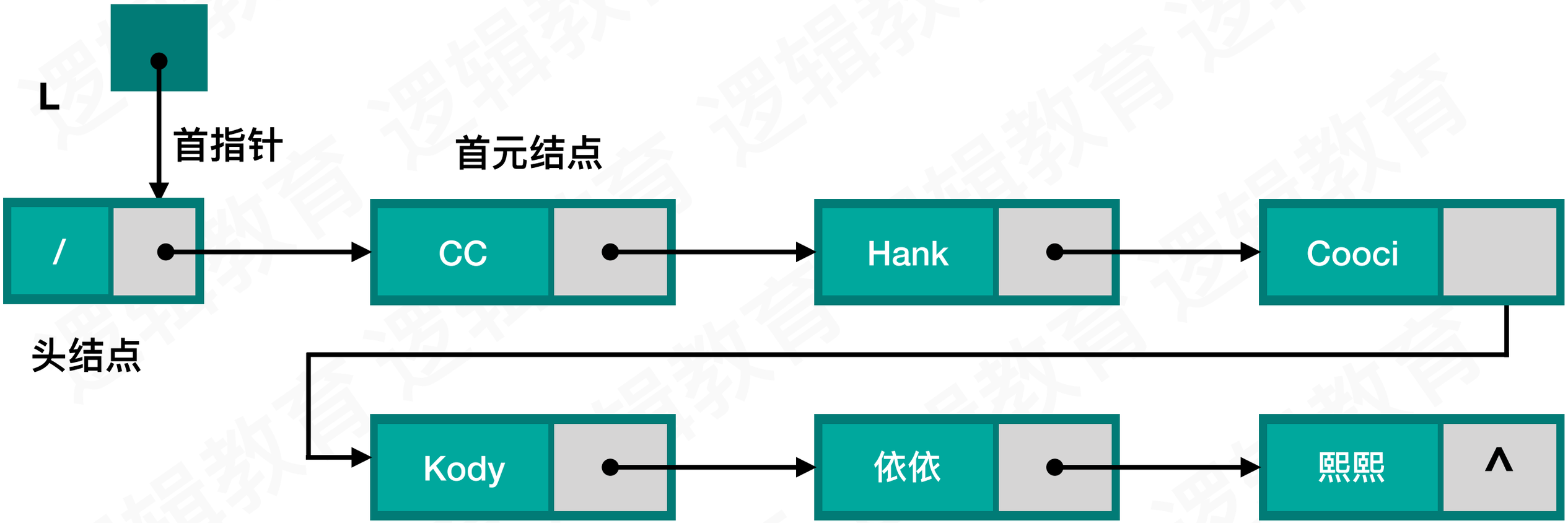
线性表—单链表逻辑状态



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

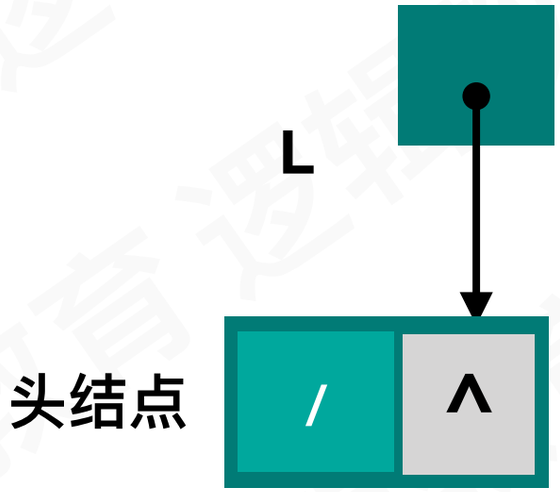
线性表—增加头结点的单链表逻辑状态



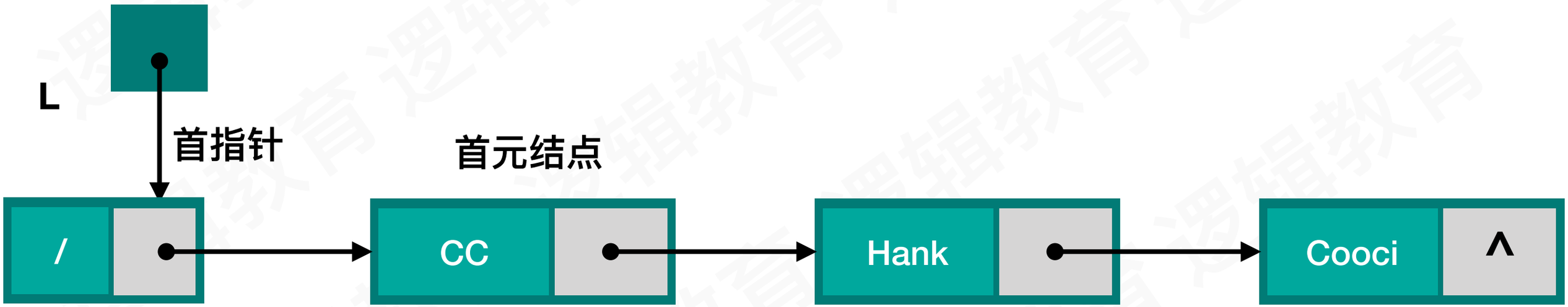
课程研发:CC老师
课程授课:CC老师

线性表—单链表为什么要增加头结点

1. 便于首元结点处理
2. 便于空表和非空表的统一处理.



(b) 空表

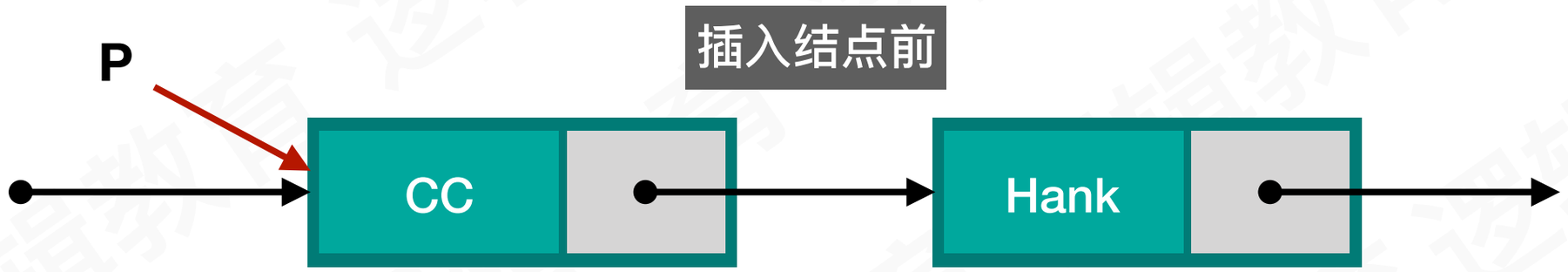


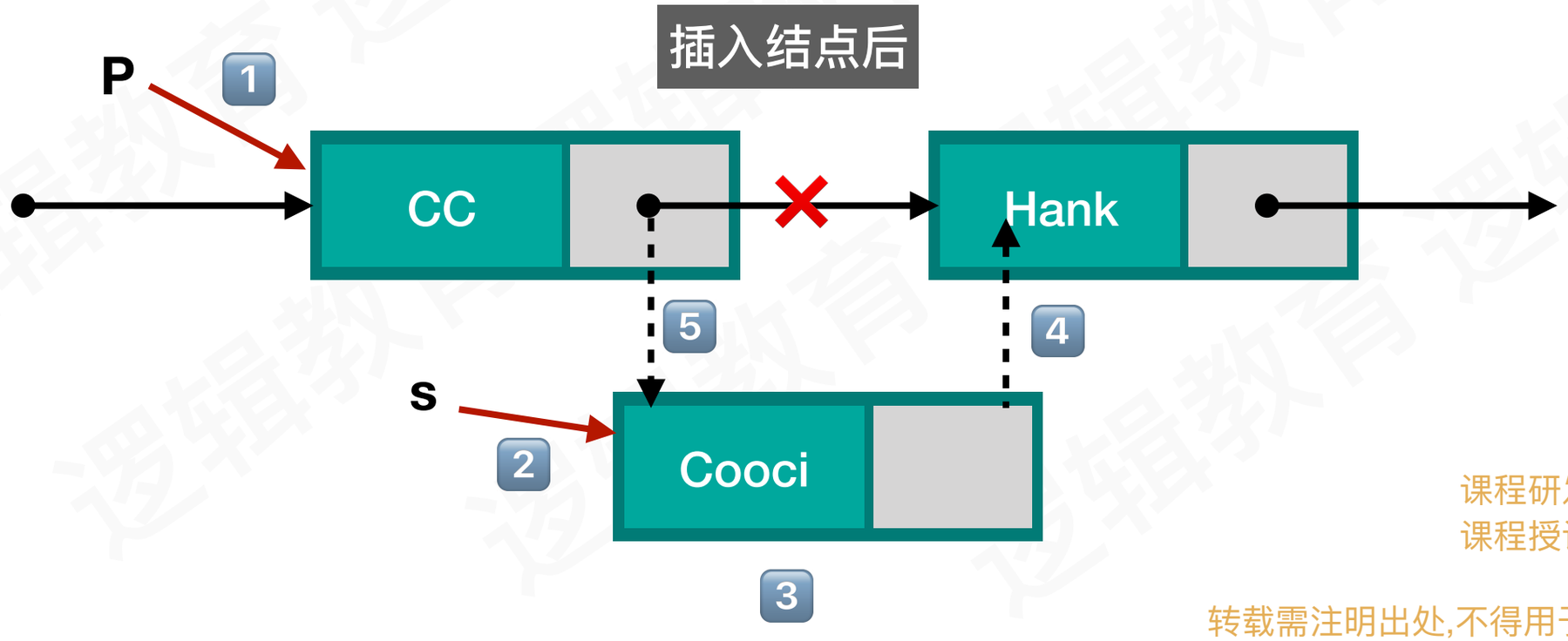
(a) 非空表

课程研发:CC老师
课程授课:CC老师

线性表—单链表插入

假设要在单链表的两个数据元素a和b之间插入一个数据元素x,已知p为其单链表存储结构中指向结点a指针.如下图所示



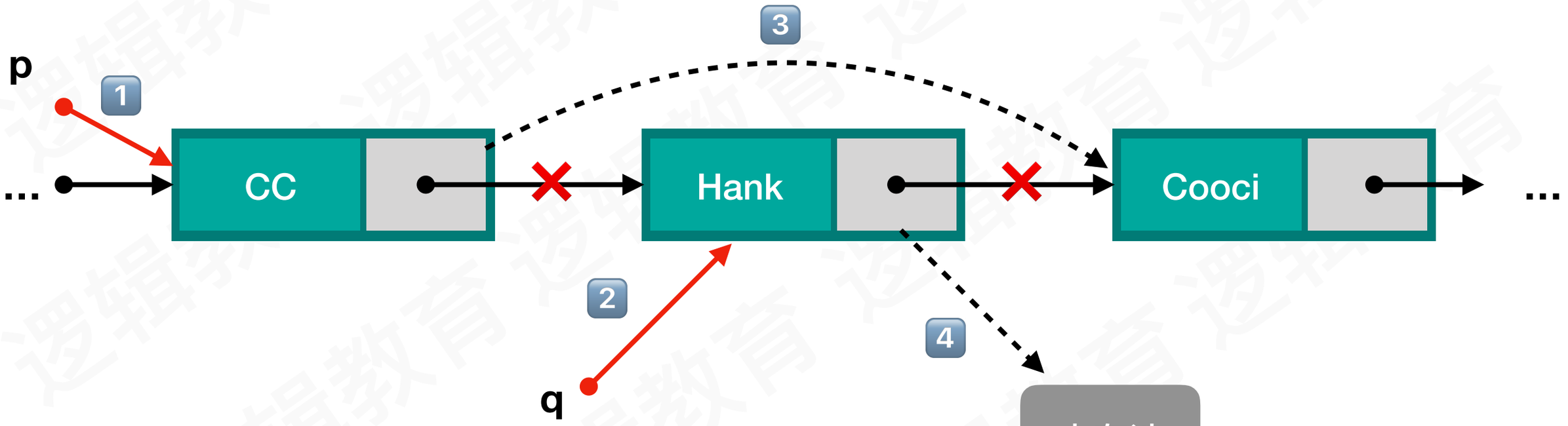


课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线性表—单链表删除

要删除单链表中指定位置的元素,同插入元素一样; 首先应该找到该位置的前驱结点; 如下图所示在单链表中删除元素Hank时,应该首先找到其前驱结点CC. 为了在单链表中实现元素CC,Hank,Cooci 之间的逻辑关系的变化,仅需修改结点CC中的指针域即可.
假设p为指向结点CC的指针

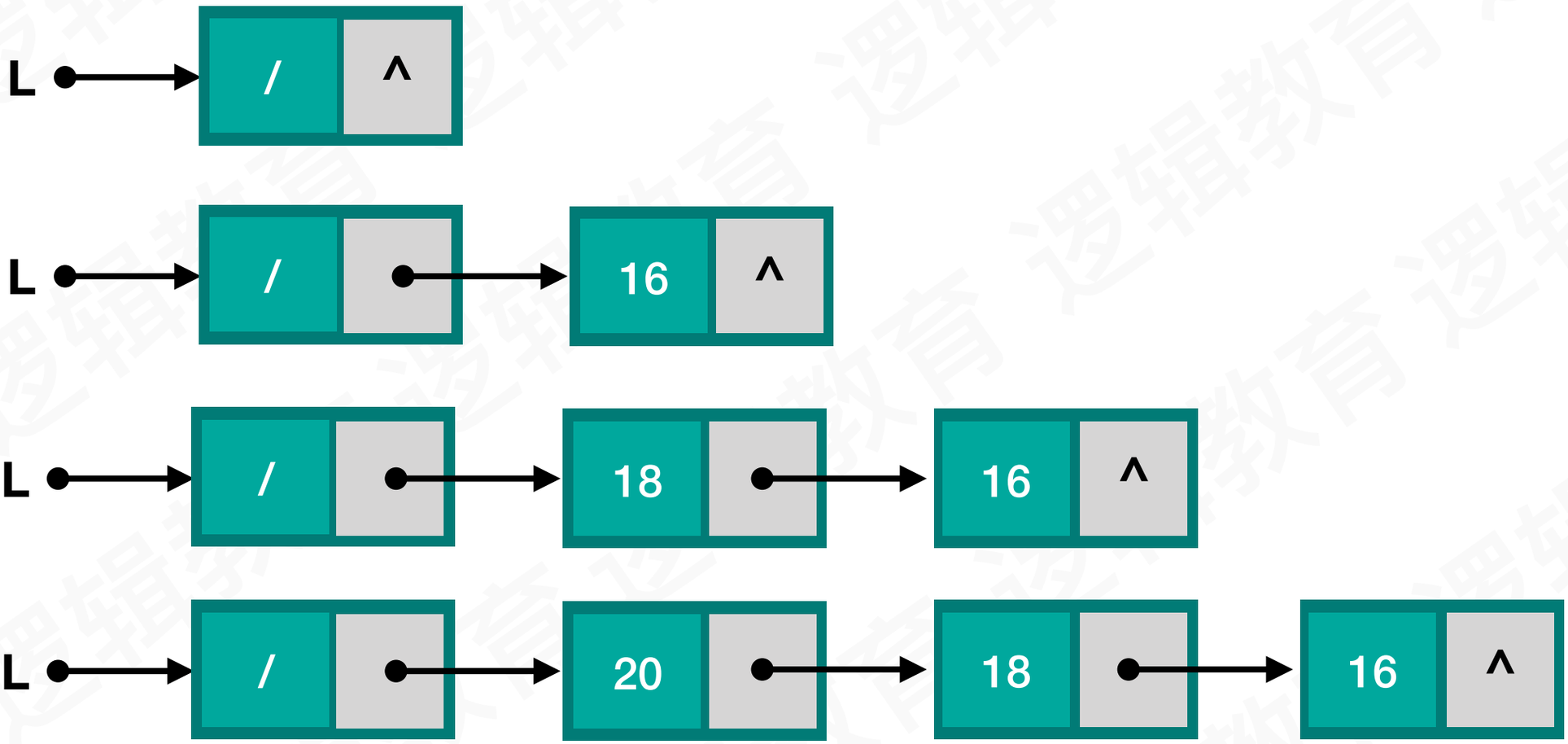


课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线性表—单链表前插法

假设插入随机数

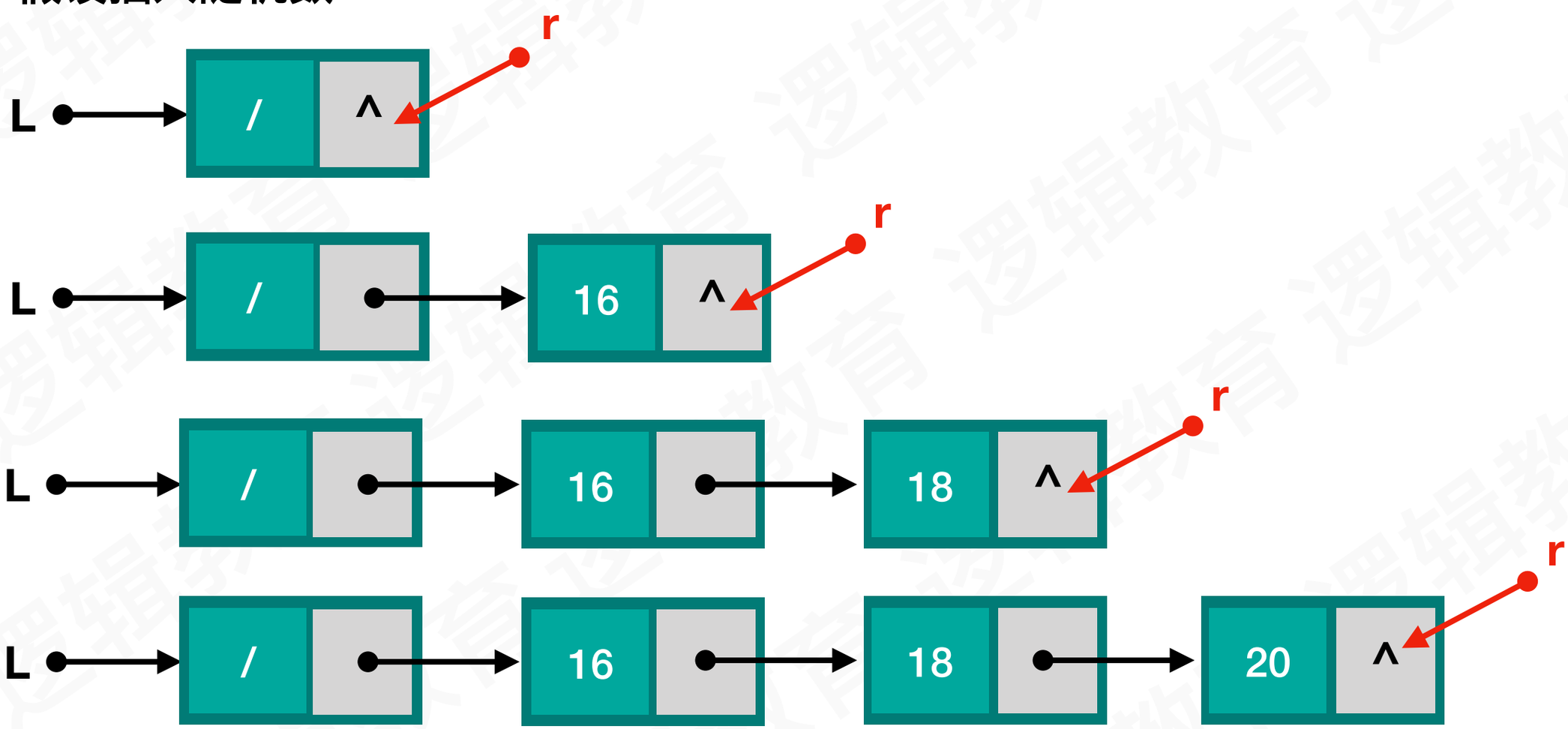


前插法创建单链表

课程研发:CC老师
课程授课:CC老师

线性表—单链表后插法

假设插入随机数



后插法创建单链表

课程研发:CC老师
课程授课:CC老师

线性表—链表结构与顺序存储结构优缺点对比

存储分配方式

- 顺序存储结构用用一段连续的存储单元依次存储线性表的数据元素;
- 单链表采用链式存储结构,用一组任意的存储单元存放线性表的元素;

时间性能

- 查找
  - 顺序存储 O(1)
  - 单链表O(n)
- 插入和删除
  - 存储结构需要平均移动一个表长一半的元素,时间O(n)
  - 单链表查找某位置后的指针后,插入和删除为 O(1)

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

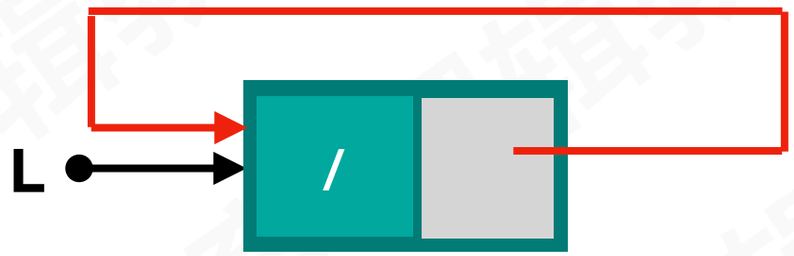
线性表—链表结构与顺序存储结构优缺点对比

空间性能

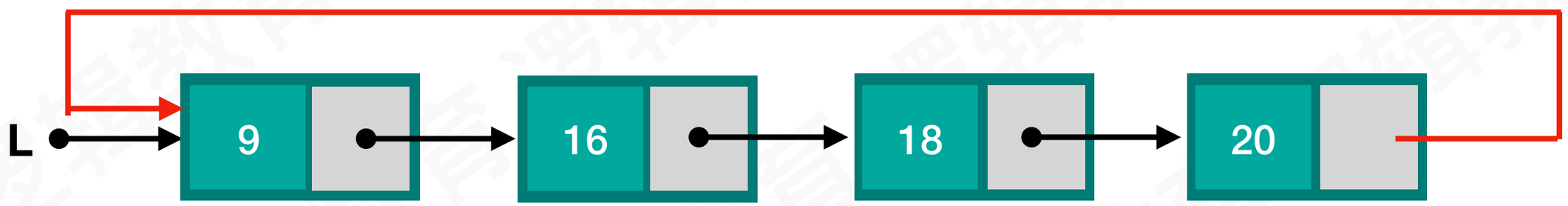
- 顺序存储结构需要预先分配存储空间,分太大,浪费空间;分小了,发生上溢出;
- 单链表不需要分配存储空间,只要有就可以分配, 元素个数也不受限制;

课程研发:CC老师
课程授课:CC老师

线性表—循环链表



(a) 空表

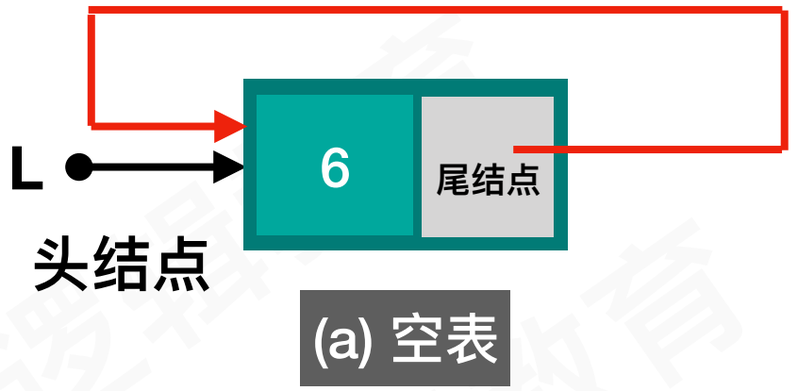


(b) 非空表

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线性表—循环链表 初始化以及赋值

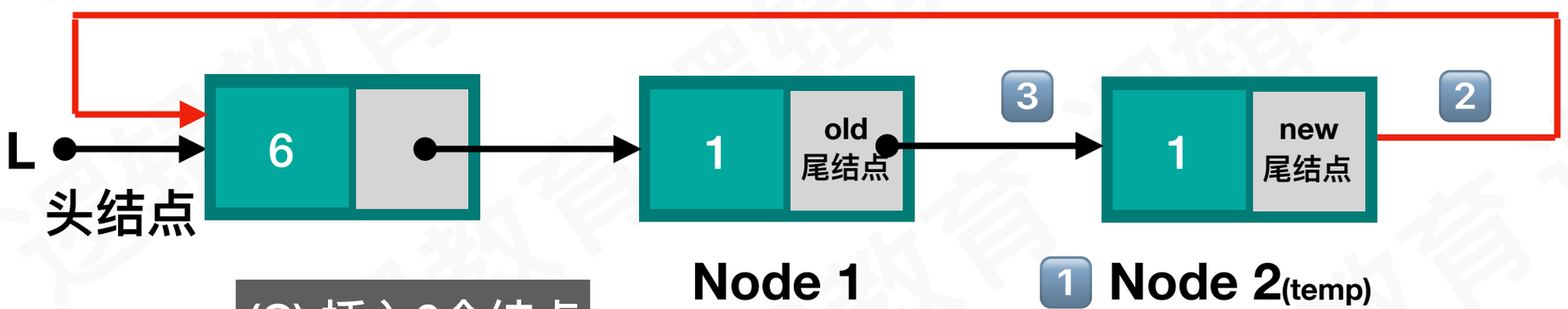


(a) 空表

```
*L = (LinkList *)malloc(sizeof(Node));
(*L)->data = item;
(*L)->next = *L;
```

(b) 插入1个结点

```
temp->data = item;
temp->next = *L;
target->next = temp
```



(C) 插入2个结点

```
temp->data = item;
temp->next = *L;
target->next = temp
```

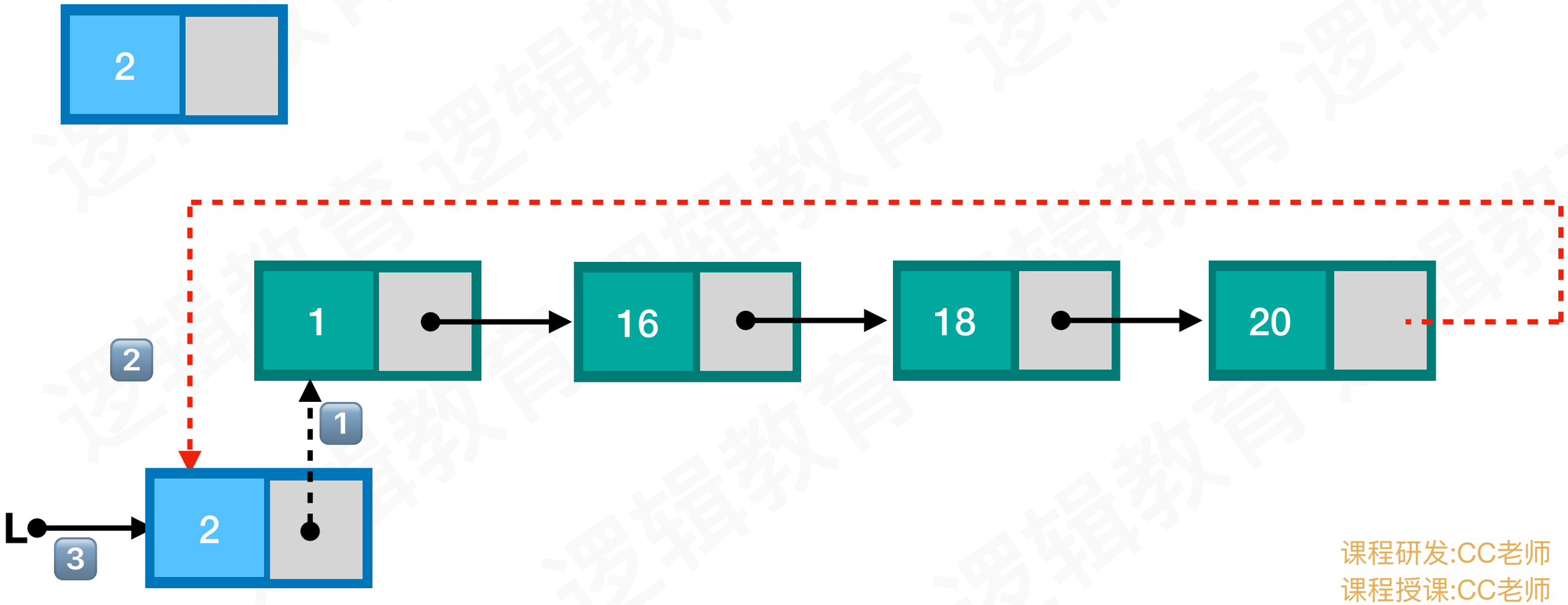

课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线性表—循环链插入数据

情况1 在插入位置在首元节点上

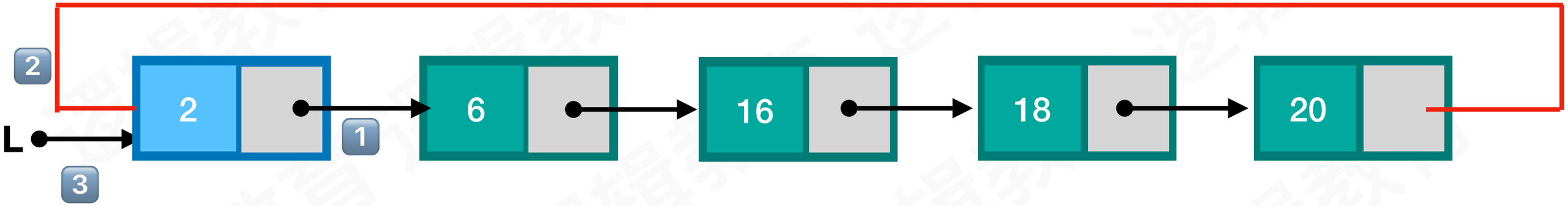
1. 判断插入位置是否在首元结点上
2. 创建新结点,并赋值给新结点



课程研发:CC老师
课程授课:CC老师

线性表—循环链插入数据

情况1 在插入位置在首元节点上



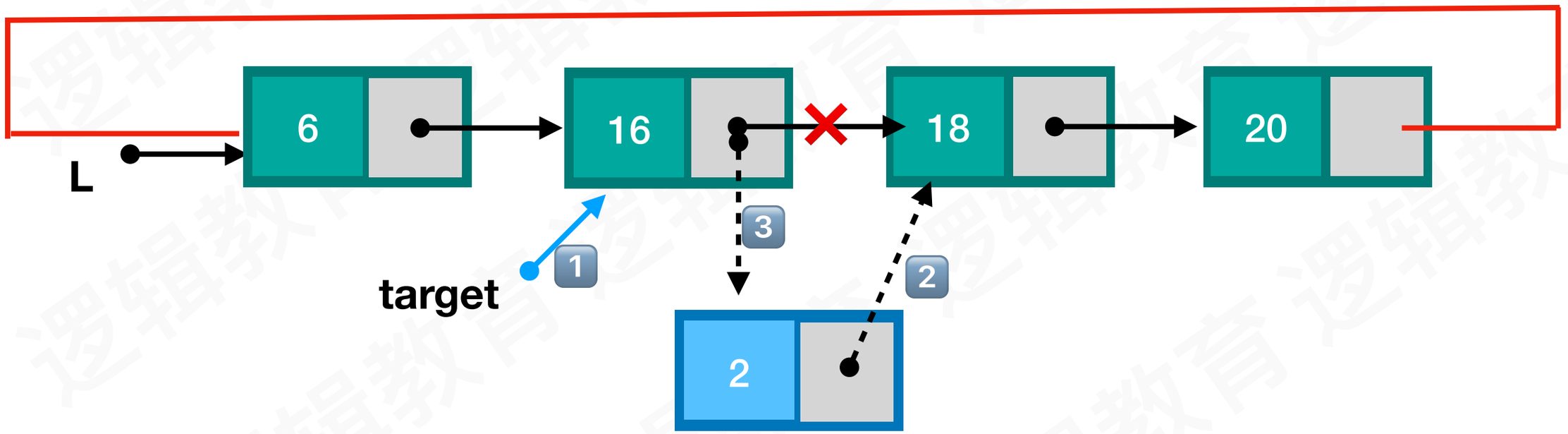
课程研发:CC老师
课程授课:CC老师

线性表—循环链插入数据

情况2 其他位置上

1. 创建新结点,并赋值给新结点

2. 找到插入的位置的前一个结点 target

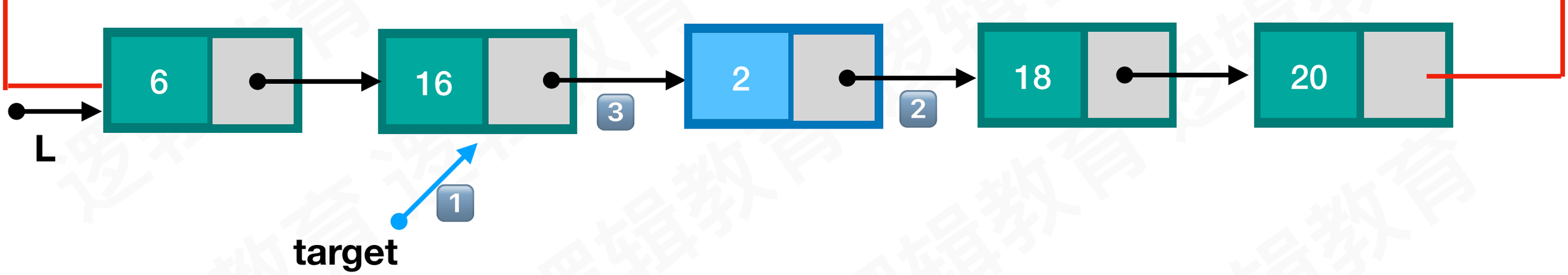


课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线性表—循环链插入数据

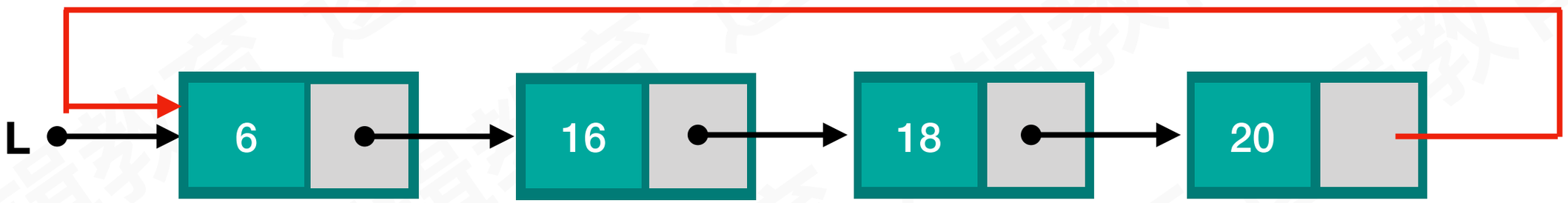
情况2 其他位置上



课程研发:CC老师
课程授课:CC老师

逻辑教育
Logic education

线性表—循环链删除数据



课程研发:CC老师
课程授课:CC老师