

列表

无序列表：插入与删除

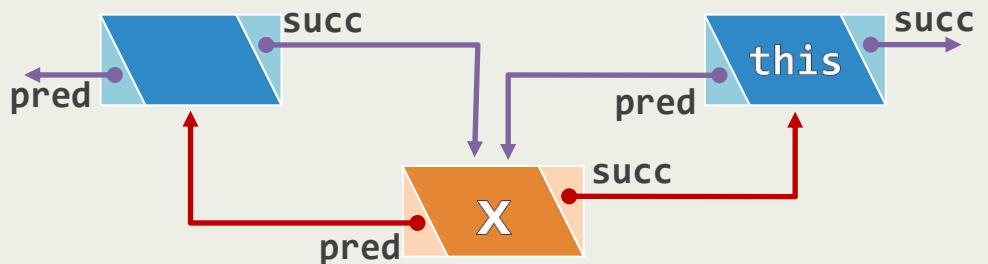
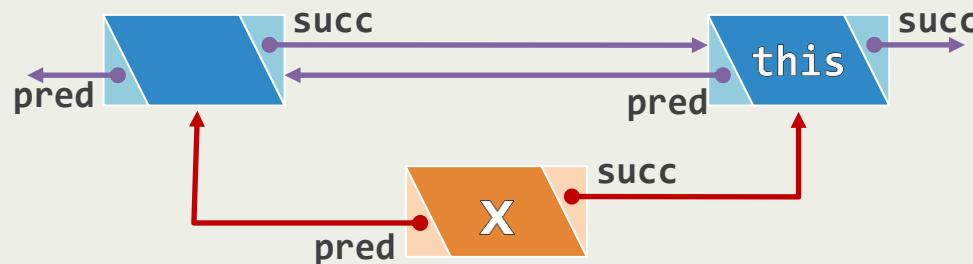
03 - C1

邓俊辉

deng@tsinghua.edu.cn

# 插入 : 思路 + 过程

❖ template <typename T> Posi(T) List<T>::insertB( Posi(T) p, T const & e )  
{ \_size++; return p->insertAsPred( e ); } //e当作p的前驱插入 ( Before )



# 插入 : 实现

❖ template <typename T> //前插入算法 ( 后插入算法完全对称 )

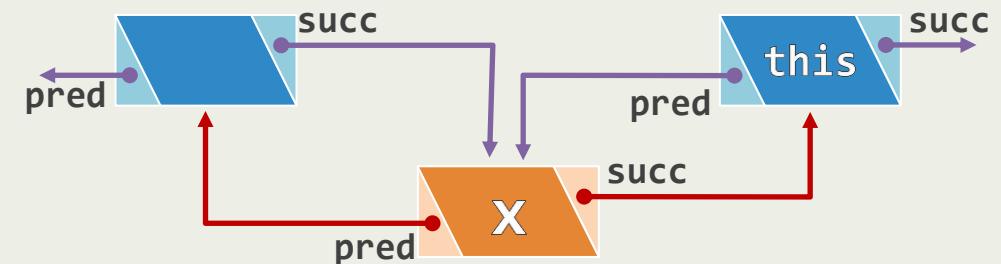
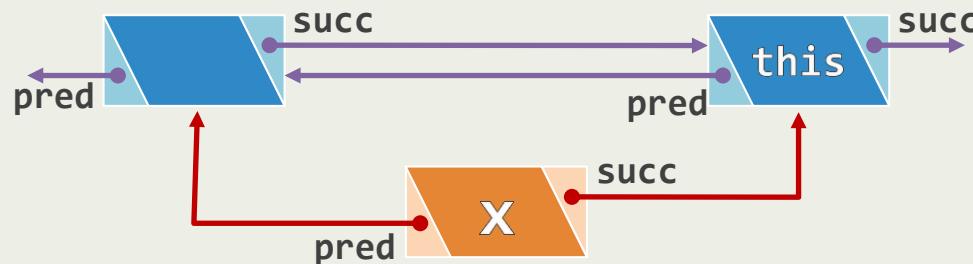
```
Posi(T) ListNode<T>::insertAsPred( T const & e ) { //O(1)
```

```
    Posi(T) x = new ListNode( e, pred, this ); //创建 ( 耗时100倍 )
```

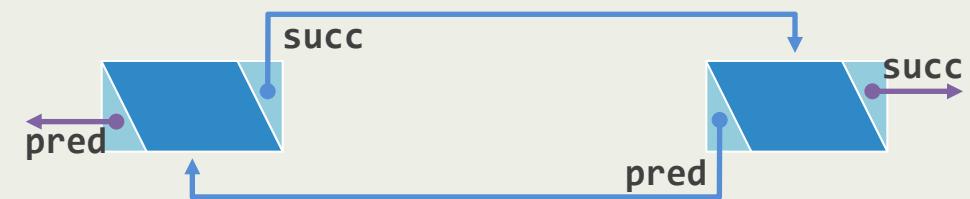
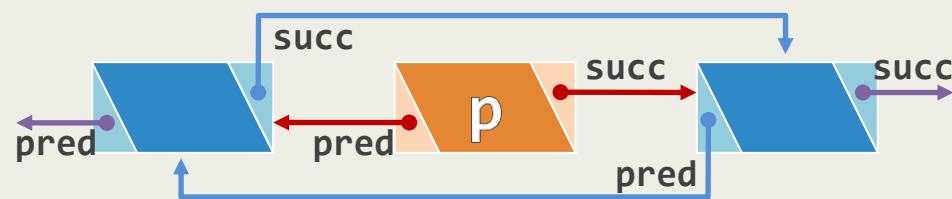
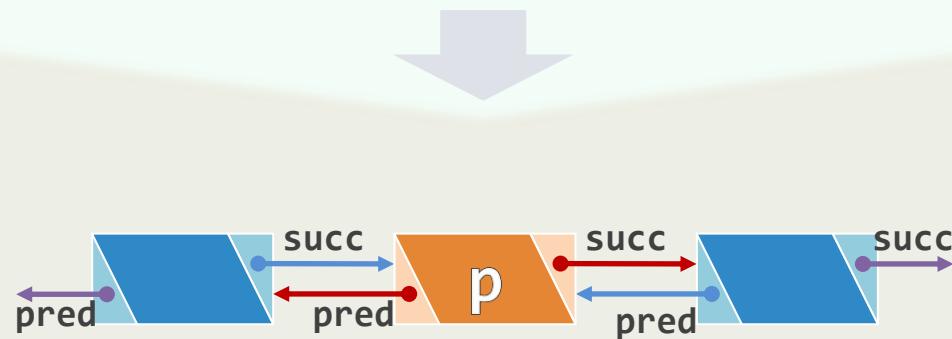
```
    pred->succ = x; pred = x; //次序不可颠倒
```

```
    return x; //建立链接 , 返回新节点的位置
```

```
} //得益于哨兵 , 即便this为首节点亦不必特殊处理——此时等效于insertAsFirst(e)
```



# 删除 : 思路 + 过程



# 删除 : 实现

❖ template <typename T> //删除合法位置p处节点，返回其数值

```
T List<T>::remove( Posi(T) p ) { //O(1)  
    T e = p->data; //备份待删除节点数值 (设类型T可直接赋值)  
  
    p->pred->succ = p->succ;  
  
    p->succ->pred = p->pred;  
  
    delete p; _size--; return e; //返回备份数值  
}
```

