03-B

列表

接口与实现

百只骆驼绕山走，九十八只在山后；尾驼露尾不见头，头驼露头出山沟

邓俊辉

deng@tsinghua.edu.cn

ListNode ADT

- 作为列表的基本元素

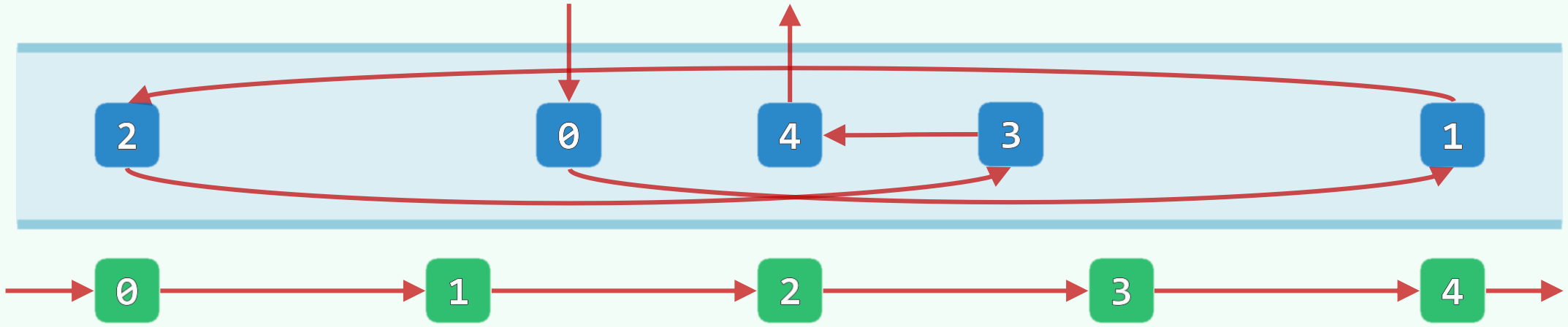
  列表节点首先需要

  独立地“封装”实现

- 列车 ~ 车厢 ~ 货物

  list ~ node ~ data

操作接口	功能
pred() / succ()	当前节点前驱/后继节点的位置
data()	当前节点所存数据对象
insertPred() / insertSucc()	插入前驱/后继节点，返回新节点位置



ListNode

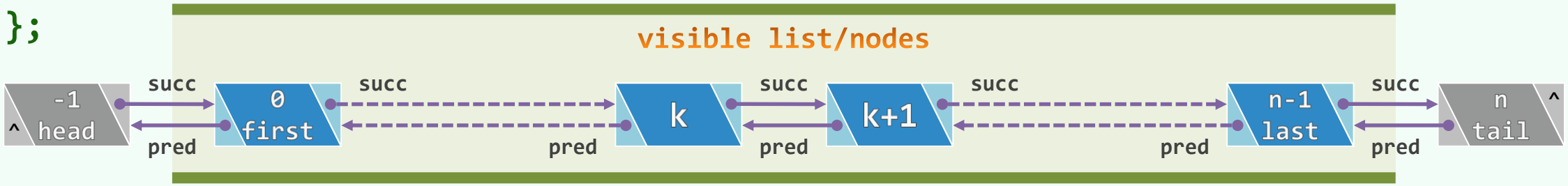
```
template <typename T> using ListNodePosi = ListNode<T>*; //列表节点位置（C++.0x）

template <typename T> struct ListNode { //简洁起见，完全开放而不再严格封装
   T data; //数值
   ListNodePosi<T> pred; //前驱
   ListNodePosi<T> succ; //后继

   ListNode() {} //针对head和tail的构造
   ListNode(T const & e, ListNodePosi<T> p = NULL, ListNodePosi<T> s = NULL)
      : data(e), pred(p), succ(s) {} //默认构造器（类T须已定义复制方法）

   ListNodePosi<T> insertPred( T const & e ); //前插入
   ListNodePosi<T> insertSucc( T const & e ); //后插入
};
```

List ADT

操作接口	功能	适用对象
size() / empty()	报告节点总数 / 判定是否为空	列表
first() / last()	返回首 / 末节点的位置	列表
insertFirst(e) / insertLast(e)	将e当作首 / 末节点插入	列表
insert(p, e), insert(e, p)	将e当作节点p的直接后继、前驱插入	列表
remove(p)	删除节点p	列表
sort(p, n) / sort()	区间 / 整体排序	列表
find(e, n, p) / search(e, n, p)	在指定区间内查找目标e	列表 / 有序列表
dedup() / uniquify()	剔除相等的节点	列表 / 有序列表
traverse(visit())	遍历列表，统一按visit()处理所有节点	列表

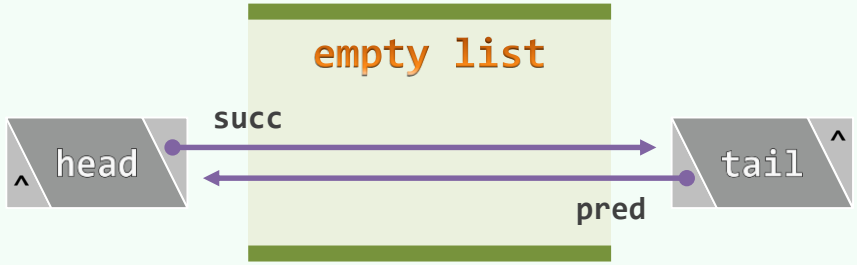
List

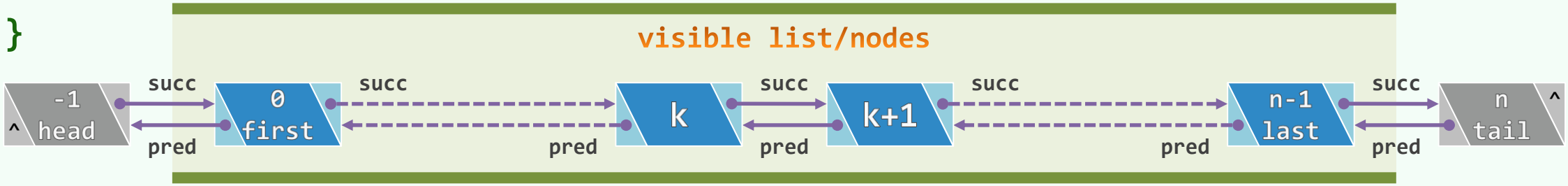
```
#include "listNode.h" //引入列表节点类

template <typename T> class List { //列表模板类

   private:    Rank _size; ListNodePosi<T> head, tail; //哨兵

               //头、首、末、尾节点的秩，可分别理解为-1、0、n-1、n

   protected:  /* ... 内部函数 */

   public:     /* ... 构造函数、析构函数、只读接口、可写接口、遍历接口 */

};
```



初始化

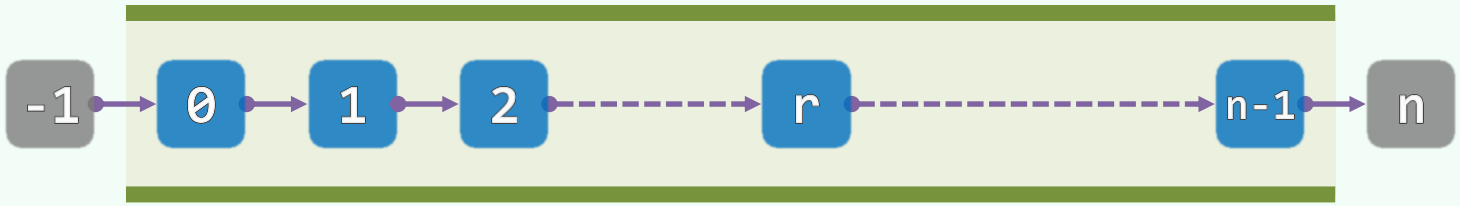
```
template <typename T> void List<T>::init() { //初始化，创建列表对象时统一调用
   head = new ListNode<T>;
   tail = new ListNode<T>;

   head->succ = tail; head->pred = NULL;
   tail->pred = head; tail->succ = NULL;

   _size = 0;
}
```





重载下标操作符，可模仿向量的循秩访问方式

```
template <typename T> //O(r)效率，虽方便，勿多用
ListNodePosi<T> List<T>::operator[]( Rank r ) const { //0 <= r < size
   ListNodePosi<T> p = first(); //从首节点出发
   while ( 0 < r-- ) p = p->succ; //顺数第r个节点即是
   return p; //目标节点
} //秩 == 前驱的总数
```



❖ **时间复杂度为** $\mathcal{O}(r)$

均匀分布时，期望复杂度为 $(1+2+3+\cdots+n)/n \;=\; \mathcal{O}(n)$