Technology Tarena 达内科技

数据结构与算法

DATASTRUCTURE DAY02

内容

上午	09:00 ~ 09:30	作业讲解和回顾
	09:30 ~ 10:20	队列
	10:30 ~ 11:20	
	11:30 ~ 12:20	
下午	14:00 ~ 14:50	链表
	15:00 ~ 15:50	
	16:00 ~ 16:50	
	17:00 ~ 17:30	总结和答疑

队列

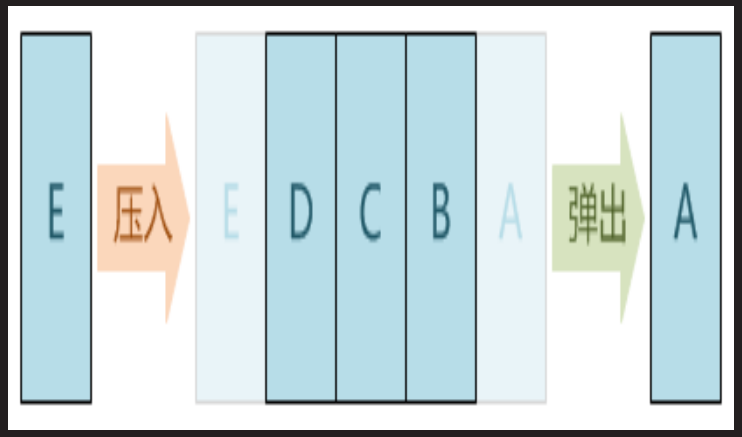
Technology Tarena 达内科技

队列的基本运算

队列的基本运算

- 队列是可以在两端增删元素的表结构，增加元素的端点称为后端，删除元素的端点称为前端
- 队列的基本运算是压入和弹出，前者相当于插入，而后者则是删除最先插入的元素，形成先进先出的运算规则
- 最先和最后插入的元素在被弹出之前可以作为队首和队尾被外界访问
- 从空队中弹出，或向满队中压入，都被认为是一种错误

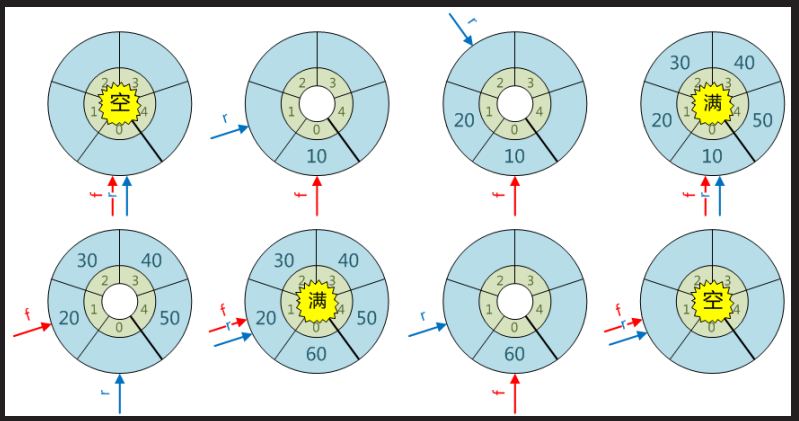
知识讲解



基于顺序表的实现

基于顺序表的实现

- 预分配空间
  - 由用户指定队列最多能容纳的元素个数，即队列容量，分配足量的内存，记下队列容量，并将前/后端指针和元素个数初始化为0
- 压入弹出
  - 被压入的元素放在后端指针处，同时后端指针上移，被弹出的元素从前端指针处获得，同时前端指针上移
- 循环使用
  - 如果前端或后端指针大于等于队列容量，且队列不空或不满，则将该指针循环复位至0
- 判空判满
  - 元素个数为0则空，禁止弹出，元素个数大于等于队列容量则满，禁止压入



知识讲解

基于顺序表的队列

【参见：TTS COOKBOOK 】

课堂练习

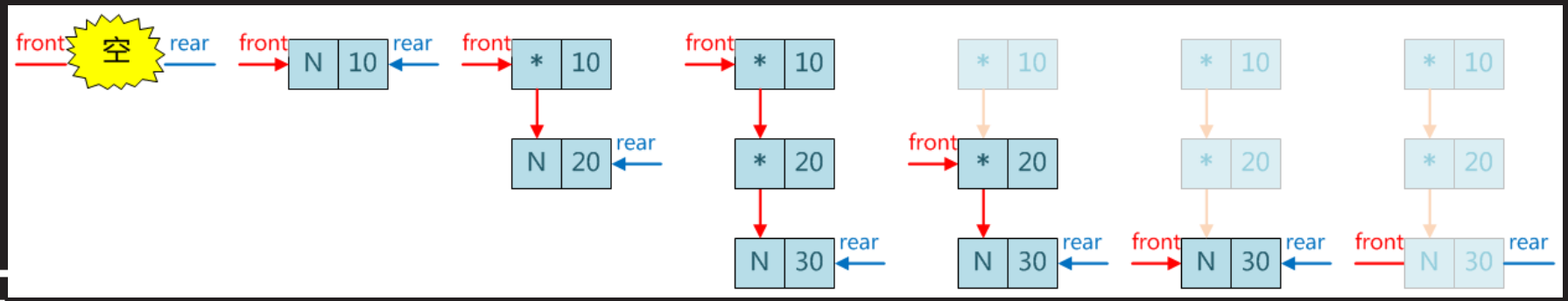
- 基于顺序表的队列

Technology Tarena 达内科技

基于链式表的实现

基于链式表的实现

- 无需预分配
  - 不需要预分配存储空间，不需要记住队列容量，也不需要判断队列是否满，随压入随分配，随弹出随释放，提升内存空间利用率
- 压入弹出
  - 被压入的元素放在新建节点中，令后端节点的后指针指向该节点，并令其成为新的后端节点，被弹出的元素由前端节点获得，释放前端节点，并令其后节点成为新的前端节点
- 注意判空
  - 前/后端指针为空表示队列已空，此时不可再弹出



Technology Tarena 达内科技

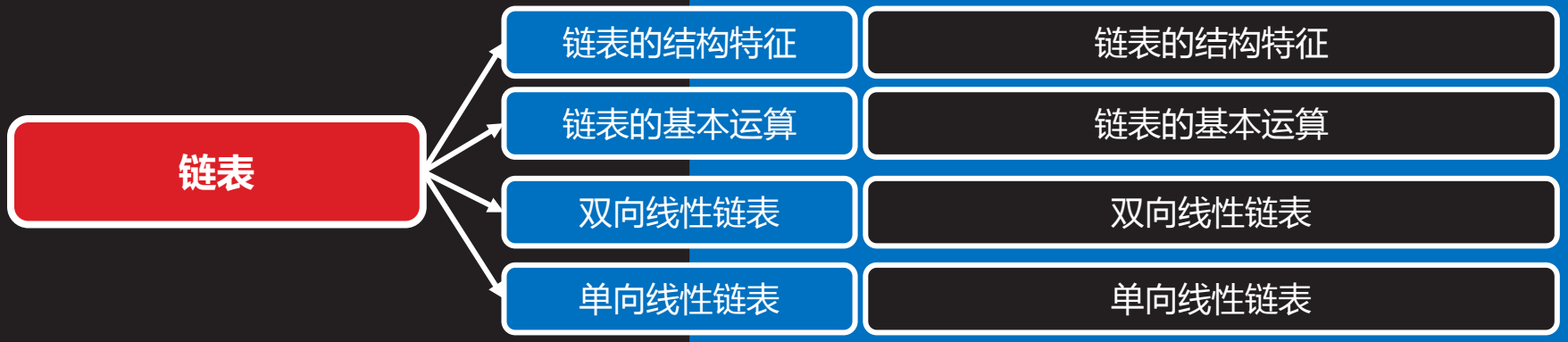
基于链式表的队列

【参见：TTS COOKBOOK 】

课堂练习

- 基于链式表的队列

链表


链表
链表的结构特征
链表的基本运算
双向线性链表
单向线性链表
链表的结构特征
链表的基本运算
双向线性链表
单向线性链表

链表的结构特征

链表的结构特征

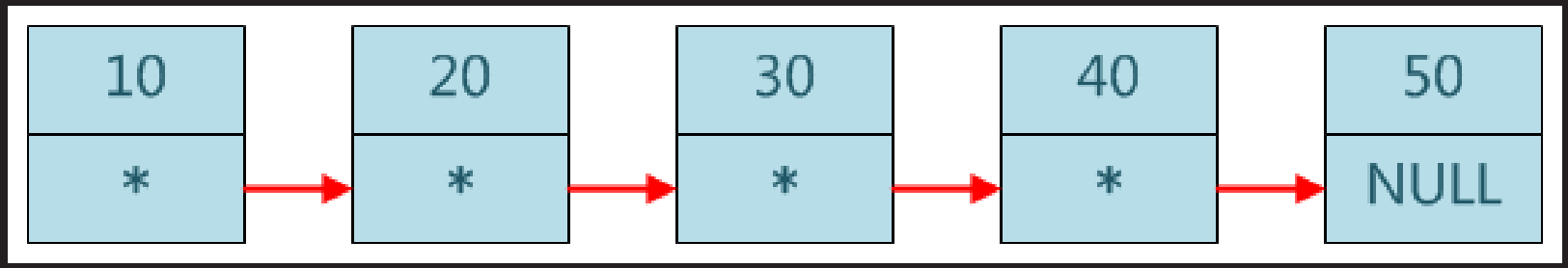
- 地址不连续的节点序列，彼此通过指针相互连接
- 根据不同的结构特征，将链表分为：
  - 单向线性链表
  - 单向循环链表
  - 双向线性链表
  - 双线循环链表
  - 数组链表
  - 链表数组
  - 二维链表

知识讲解

链表的结构特征（续1）

- 单向线性链表
  - 每个节点除了存放元素数据以外，还需要保存指向下一个节点的指针，即所谓后指针
  - 链表尾节点的后指针为空指针

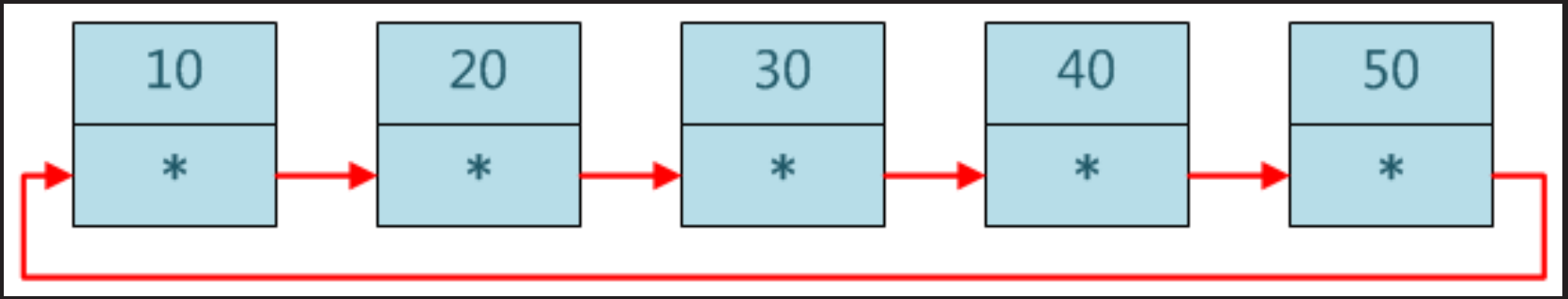
知识讲解



链表的结构特征（续2）

- 单向循环链表
  - 每个节点除了存放元素数据以外，还需要保存指向下一个节点的指针，即所谓后指针
  - 链表尾节点的后指针指向首节点，首尾相接构成环状

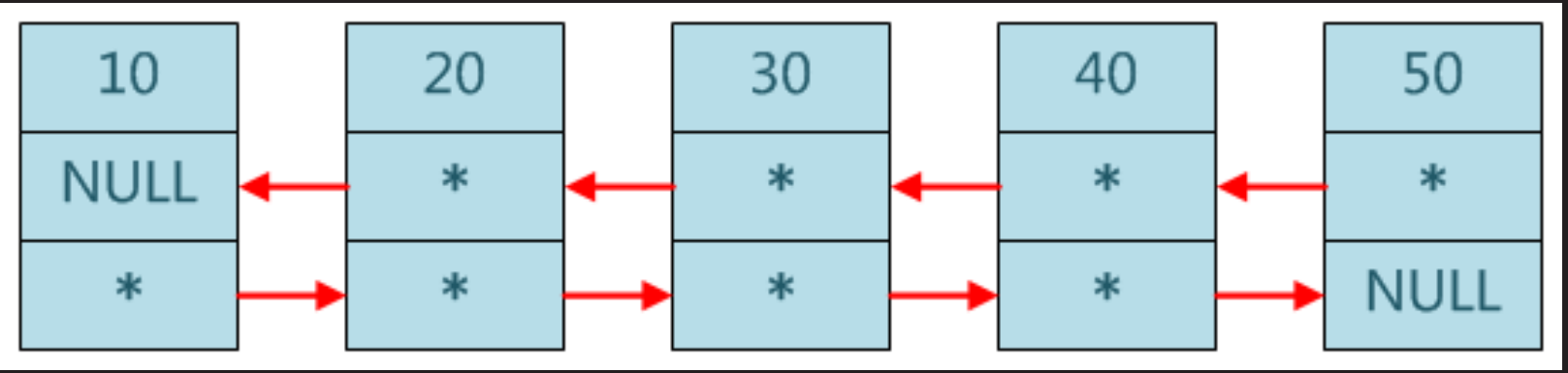
知识讲解



链表的结构特征（续3）

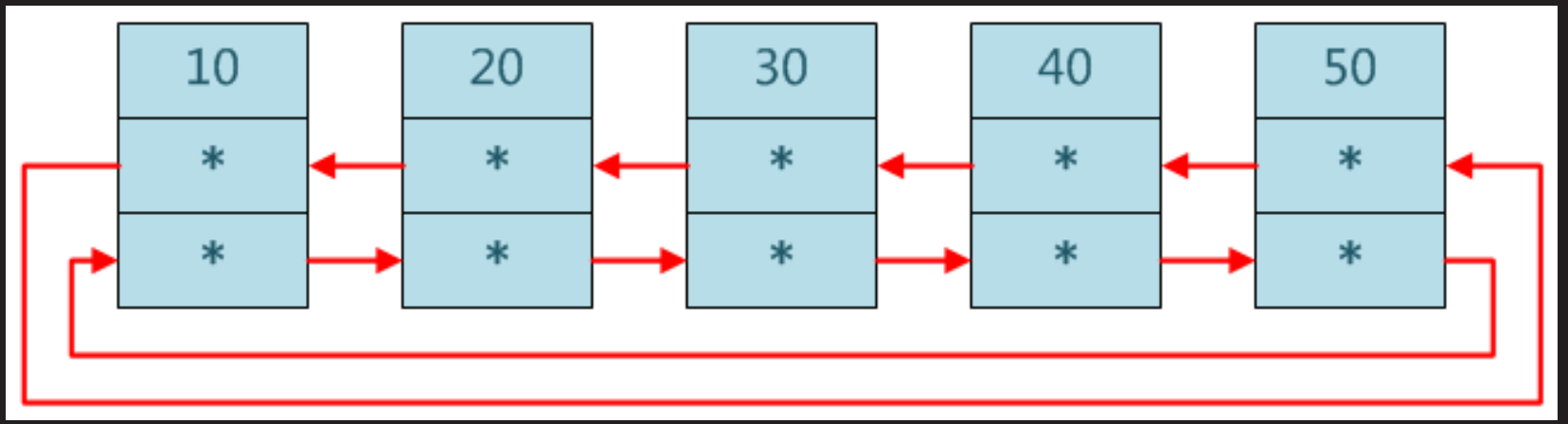
- 双向线性链表
  - 每个节点除了存放元素数据以外，还需要保存指向下一个节点的指针，即所谓后指针，以及指向前一个节点的指针，即所谓前指针
  - 链表首节点的前指针和尾节点的后指针俱为空指针

知识讲解



链表的结构特征（续4）

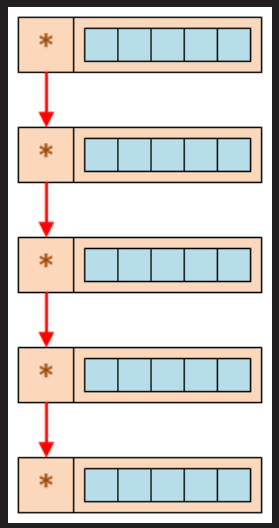
- 双向循环链表
  - 每个节点除了存放元素数据以外，还需要保存指向下一个节点的指针，即所谓后指针，以及指向前一个节点的指针，即所谓前指针
  - 链表首节点的前指针和尾节点的后指针分别指向链表的尾节点和首节点

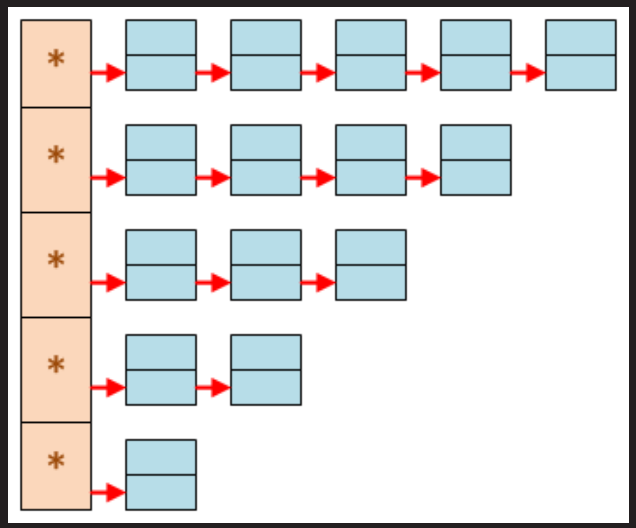


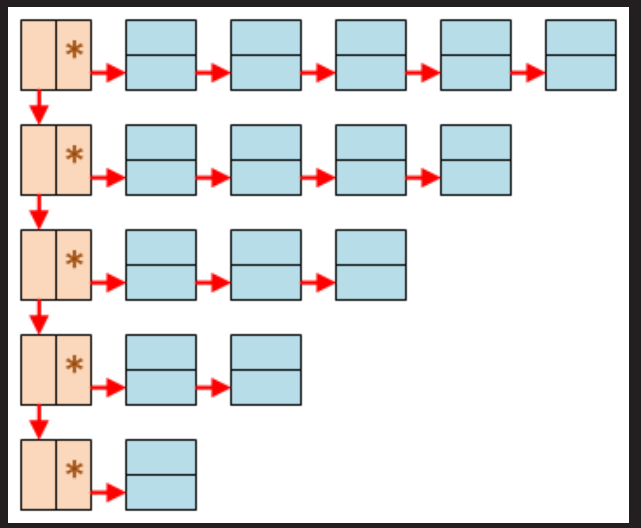
链表的结构特征（续5）

- 数组链表
  - 链表中的每个元素都是数组，即由数组构成的链表
- 链表数组
  - 数组中的每个元素都是链表，即由链表构成的数组
- 二维链表
  - 链表中的每个元素都是链表，即由链表构成的链表

知识讲解







Technology Tarena 达内科技

链表的基本运算

链表的基本运算

- 追加
  - 在链表的尾部添加新元素
- 插入
  - 在特定位置的元素之前或之后加入新元素
- 删除
  - 删除特定位置的元素
- 遍历
  - 依次访问链表中的每一个元素，不重复，不遗漏

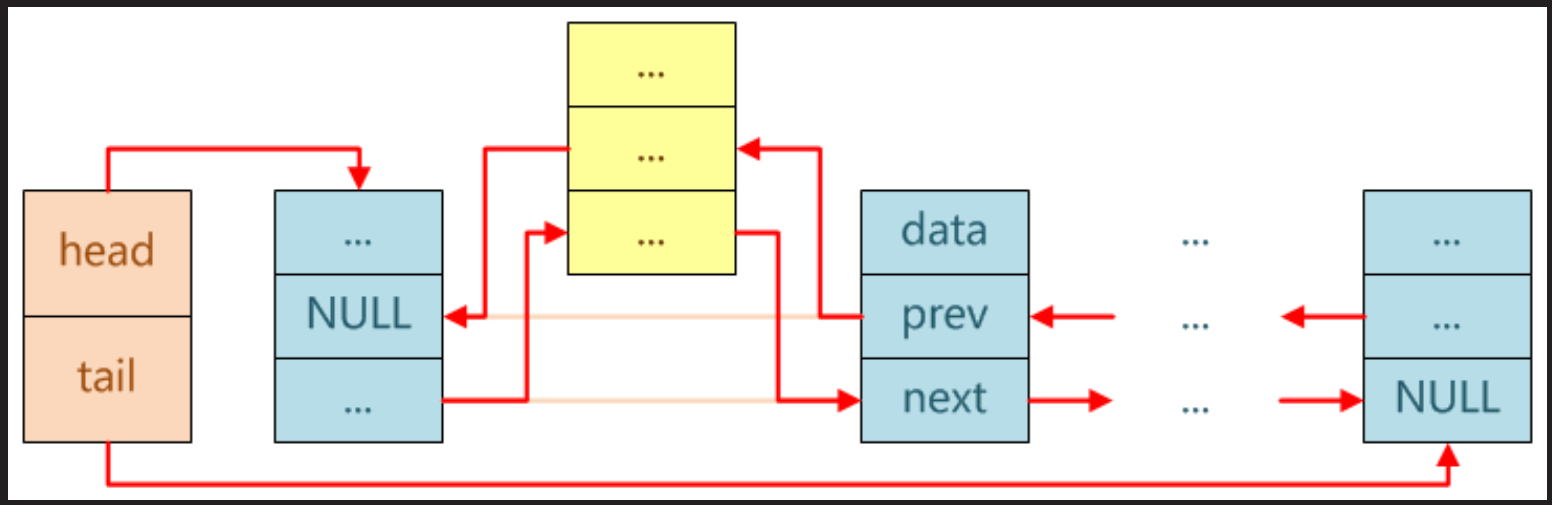
知识讲解

双向线性链表

双向线性链表

- 在给定节点之前插入
  - 创建持有待插入元素的新节点，令其前指针指向给定节点的前节点，令其后指针指向给定节点
  - 若新建节点存在前节点，则令其前节点的后指针指向新建节点，否则新建节点为新的首节点
  - 令新建节点的后节点的前指针指向新建节点

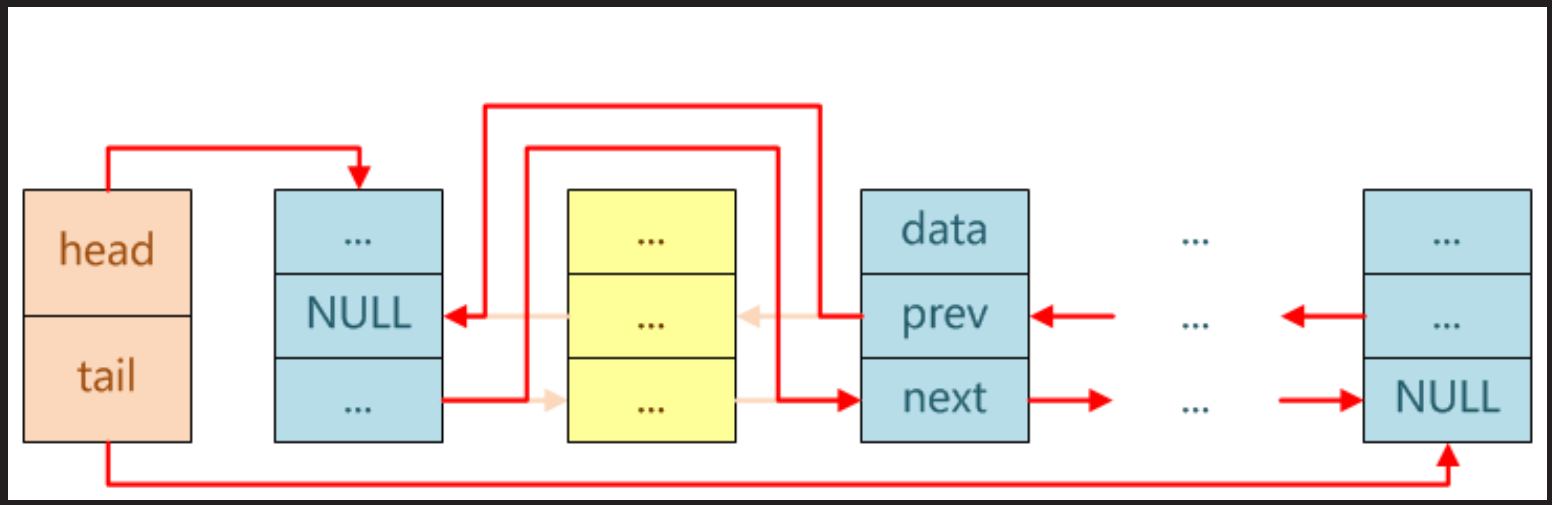
知识讲解



双向线性链表（续1）

- 删除节点
  - 若将亡节点存在前节点，则令其前节点的后指针指向将亡节点的后节点，否则将亡节点的后节点为新的首节点
  - 若将亡节点存在后节点，则令其后节点的前指针指向将亡节点的前节点，否则将亡节点的前节点为新的尾节点
  - 销毁将亡节点

知识讲解



双向线性链表

【参见：TTS COOKBOOK 】

课堂练习

- 双向线性链表

Technology Tarena 达内科技

单向线性链表

单向线性链表

- 追加
  - void list_append (LIST* list, int data) {

```
        LIST_NODE* node = create_node (data);
        if (list->tail)
            list->tail->next = node;
        else
            list->head = node;
        list->tail = node;
    }
```


单向线性链表（续1）

- 测长
  - size_t list_size (LIST* list) {

```
        size_t size = 0;
        LIST_NODE* node = NULL;
        for (node = list->head; node;
            node = node->next)
            ++size;
        return size;
    }
```

知识讲解

单向线性链表（续2）

- 正向打印
  - void list_print (LIST* list) {

```
        LIST_NODE* node = NULL;
        for (node = list->head; node;
            node = node->next)
            printf ("%d ", node->data);
        printf ("\n");
    }
```

知识讲解

单向线性链表（续3）

- 反向打印
  - void rprint (LIST_NODE* head) {
    if (head) {
    rprint (head->next);
    printf ("%d ", head->data);
    }
    }
  - void list_rprint (LIST* list) {
    rprint (list->head);
    printf ("\n");
    }

单向线性链表（续4）

- 逆转
  - void reverse (LIST_NODE* head) {
    if (head && head->next) {
    reverse (head->next);
    head->next->next = head;
    head->next = NULL;
    }
    }
  - void list_reverse (LIST* list) {
    reverse (list->head);
    LIST_NODE* swap = list->head;
    list->head = list->tail;
    list->tail = swap; }

单向线性链表（续5）

- 中位
  -
```
int list_middle (LIST* list) {
    LIST_NODE* slow = NULL, *fast = NULL;
    for (slow = fast = list->head;
        fast->next && fast->next->next;
        fast = fast->next->next)
        slow = slow->next;
    return slow->data;
}
```

知识讲解

单向线性链表（续6）

- 有序合并
  - LIST_NODE* merge (LIST_NODE* head1,
    LIST_NODE* head2) {
    if (! head1) return head2;
    if (! head2) return head1;
    LIST_NODE* head = NULL;
    if (head1->data < head2->data)
    (head = head1)->next = merge (
    head1->next, head2);
    else
    (head = head2)->next = merge (
    head2->next, head1);
    return head; }

知识讲解

单向线性链表（续7）

- 有序合并
  - void list_merge (LIST* list1,
    LIST* list2) {
    list1->head = merge (
    list1->head, list2->head);
    if (list1->tail->data <
    list2->tail->data)
    list1->tail = list2->tail;
    list2->head = list2->tail = NULL;
    }

知识讲解

单向线性链表（续8）

- 判环
  -
```
int list_ring (LIST_NODE* node) {
    LIST_NODE* slow = node;
    LIST_NODE* fast = slow->next;
    while (fast && fast->next &&
        fast->next->next) {
        if (slow == fast)
            return 1;
        slow = slow->next;
        fast = fast->next->next;
    }
    return 0;
}
```

知识讲解

单向线性链表（续9）

- 删除非尾节点
  - void list_erase (LIST_NODE* node) {
    node->data = node->next->data;
    node->next = destroy_node (node->next);
    }

知识讲解

单向线性链表

【参见：TTS COOKBOOK 】

课堂练习

- 单向线性链表

Technology Tarena 达内科技

总结和答疑