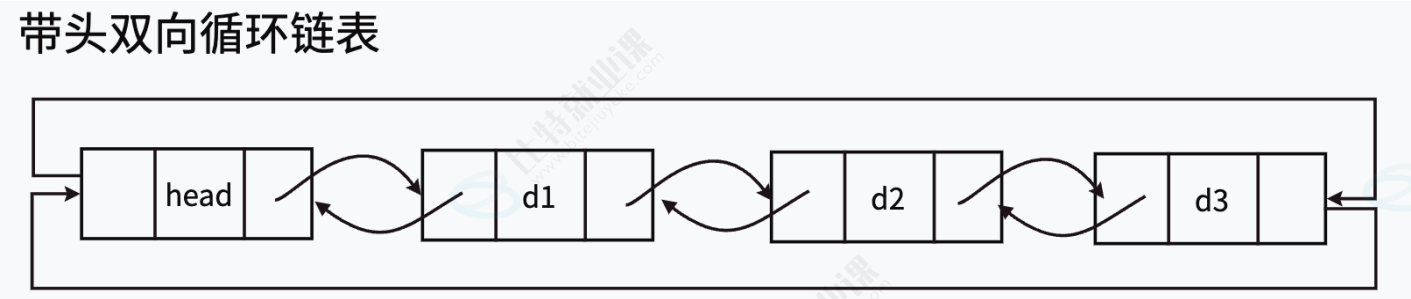
第30讲：双向链表专题

目录

正文开始

1. 双向链表的结构



> 注意：这里的“带头”跟前面我们说的“头节点”是两个概念，实际前面的在单链表阶段称呼不严谨，但是为了同学们更好的理解就直接称为单链表的头节点。
>
> 带头链表里的头节点，实际为“哨兵位”，哨兵位节点不存储任何有效元素，只是站在这里“放哨的”
>
> “哨兵位”存在的意义：
>
> 遍历循环链表避免死循环。

2. 双向链表的实现

```
typedef int LTDataType;
typedef struct ListNode
{
    struct ListNode* next; //指针保存下一个节点的地址
    struct ListNode* prev; //指针保存前一个节点的地址
    LTDataType data;
}LTNode;

```

```
//void LTInit(LTNode** pphead);
LTNode* LTInit();
void LTDestroy(LTNode* phead);
void LTPrint(LTNode* phead);
bool LTEmpty(LTNode* phead);

void LTPushBack(LTNode* phead, LTDataType x);
void LTPopBack(LTNode* phead);

void LTPushFront(LTNode* phead, LTDataType x);
void LTPopFront(LTNode* phead);
//在pos位置之后插入数据
void LTInsert(LTNode* pos, LTDataType x);
void LTErase(LTNode* pos);
LTNode *LTFind(LTNode* phead,LTDataType x);
```

3. 顺序表和双向链表的优缺点分析

不同点	顺序表	链表（单链表）
存储空间上	物理上一定连续	逻辑上连续，但物理上不一定连续
随机访问	支持O(1)	不支持：O(N)
任意位置插入或者删除元素	可能需要搬移元素，效率低O(N)	只需修改指针指向
插入	动态顺序表，空间不够时需要扩容	没有容量的概念
应用场景	元素高效存储+频繁访问	任意位置插入和删除频繁

完