数据结构_带头双向循环链表

List.h

相较于之前的顺序表和单向链表，双向链表的逻辑结构稍微复杂一些，但是在实现各种接口的时候是很简单的。因为不用找尾，写起来会舒服一点。（也可能是因为最近一直在写这个的原因）

```
#pragma once
#include<stdio.h>
#include<assert.h>
#include<stdlib.h>
#include<stdbool.h>

typedef int LTDataType;
typedef struct ListNode
{
    struct ListNode* prev;
    LTDataType data;
    struct ListNode* next;

}LTNode;

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);
```

List.c

在实现接口的时候，除了没有找尾，其他的操作和单向链表是差不多的，这里就不多说了。**(注意代码里的注释)**

```c
#define _CRT_SECURE_NO_WARNINGS 1

#include"List.h"

LTNode* BuyNode(LTDataType x)
{
    LTNode* node = (LTNode*)malloc(sizeof(LTNode));
    if (node == NULL)
    {
        perror("malloc is fail!");
        exit(1);
    }
    node->data = x;
    node->prev = node->next = node;
    return node;
}

LTNode* LTInit()
{
    LTNode* phead = BuyNode(-1);
    return phead;
}

void LTPrint(LTNode* phead)
{
    assert(phead);
    LTNode* pcur = phead->next;
    while (pcur != phead)
    {
        printf("%d->",pcur->data);
        pcur = pcur->next;
    }
    printf("\n");//注意换行
}

void LTPushBack(LTNode* phead, LTDataType x)
{
    assert(phead);
    LTNode* newnode = BuyNode(x);
    newnode->prev = phead->prev;
    newnode->next = phead;

    newnode->prev->next = newnode;
    phead->prev = newnode;
}

void LTPopBack(LTNode* phead)
{
    assert(phead && phead->next != phead);//第二个很容易忽视!
    LTNode* del = phead->prev;
    del->prev->next = phead;
    phead->prev = del->prev;
    free(del);
```

```
    del = NULL;
}

void LTPushFront(LTNode* phead, LTDataType x)
{
    assert(phead);
    LTNode* newnode = BuyNode(x);
    phead->next->prev = newnode;
    newnode->next = phead->next;

    phead->next = newnode;
    newnode->prev = phead;
}

void LTPopFront(LTNode* phead)
{
    assert(phead && phead->next != phead);
    LTNode* del = phead->next;
    del->next->prev = phead;
    phead->next = del->next;
    free(del);
    del = NULL;
}

LTNode* LTFind(LTNode* phead, LTDataType x)
{
    assert(phead && phead->next != phead);
    LTNode* pcur = phead->next;
    while (pcur != phead)
    {
        if (pcur->data == x)//别忘了连等号！！！
            return pcur;
        pcur = pcur->next;
    }
    return NULL;
}

void LTErase(LTNode* pos)
{
    assert(pos);
    pos->next->prev = pos->prev;
    pos->prev->next = pos->next;
    free(pos);
    pos = NULL;
}

void LTInsert(LTNode* pos, LTDataType x)
{
    assert(pos);
    LTNode* newnode = BuyNode(x);
    pos->next->prev = newnode;
    newnode->next = pos->next;
    newnode->prev = pos;
    pos->next = newnode;
```

```
}

void LTDestroy(LTNode* phead)//理论上这里应该传二级指针，但是为了保持接口的一致性，用一级。
{
    assert(phead);
    LTNode* des = phead->next;
    while (des != phead)
    {
        LTNode* next = des->next;
        free(des);
        des = next;
    }
    free(phead);
    phead = des = NULL;
}
```

本博客旨在记录学习过程，以后忘了随时来看。