**南京师范大学中北学院**

**《数据结构》**

**实**

**验**

**报**

**告**

**姓 名: 钟玮**

**学 号: 85213241**

**班 级: 计算机二班**

**日 期: 2024年3月20日**

**双向循环链表的基本操作**

1. **实验目的**

**1、掌握双向循环链表的定义；**

**2、掌握双向循环链表的基本操作，如初始化、查找、插入、删除等。**

1. **实验内容**

**1、初始化一个带头结点的双向循环链表；**

**2、获取带头结点的双向循环链表的第i个节点，并返回；**

**3、在带头结点的双向循环链表L的第i个元素之前插入元素e；**

**4、删除带头结点的双向循环链表L的第i个元素；**

**5、遍历带头结点的双向循环链表中的元素，并打印。**

1. **实验主要代码**

**可以截图并做简要解释，如：**

* 1. **初始化一个带头结点的双向循环链表**

**typedef struct DuLNode**

**{**

**ElemType data;**

**struct DuLNode \*prior;**

**struct DuLNode \*next;**

**} DuLNode, \*DuLinkList;**

**Status ListInit\_DuL(DuLinkList &L)**

**{**

**L = (DuLinkList)malloc(sizeof(DuLNode));**

**if (!L)**

**exit(OVERFLOW);**

**L->prior = L;**

**L->next = L;**

**return OK;**

**}**

* 1. **获取带头结点的双向循环链表的第i个节点，并返回**

**DuLinkList GetElemP\_DuL(DuLinkList va, int i)**

**{**

**DuLinkList p;**

**p = va->next;**

**int j = 1;**

**while (p != va && j < i)**

**{**

**p = p->next;**

**++j;**

**}**

**if (p == va && j < i)**

**return NULL;**

**else**

**return p;**

**}**

* 1. **在带头结点的双向循环链表L的第i个元素之前插入元素e**

**Status ListInsert\_DuL(DuLinkList &L, int i, ElemType e)**

**{**

**DuLinkList p, s;**

**if (!(p = GetElemP\_DuL(L, i)))**

**return ERROR;**

**if (!(s = (DuLinkList)malloc(sizeof(DuLNode))))**

**return ERROR;**

**s->data = e;**

**s->prior = p->prior;**

**p->prior->next = s;**

**s->next = p;**

**p->prior = s;**

**return OK;**

**}**

* 1. **删除带头结点的双向循环链表L的第i个元素**

**Status ListDelete\_DuL(DuLinkList &L, int i, ElemType &e)**

**{**

**DuLinkList p;**

**if (!(p = GetElemP\_DuL(L, i)))**

**return ERROR;**

**e = p->data;**

**p->prior->next = p->next;**

**p->next->prior = p->prior;**

**free(p);**

**return OK;**

**}**

* 1. **遍历带头结点的双向循环链表中的元素，并打印**

**void ListTraverse\_DuL(DuLinkList &L)**

**{**

**DuLinkList p;**

**p = L->next;**

**printf("当前双向循环链表的元素为：\n");**

**while (p != L)**

**{**

**printf("%d ", p->data);**

**p = p->next;**

**}**

**printf("\n");**

**}**

**6、main函数**

**int main()**

**{**

**DuLinkList L;**

**ElemType e, e1, e2;**

**int loc1, loc2;**

**ListInit\_DuL(L);**

**for (int i = 1; i <= 6; i++)**

**{**

**e = rand() % 50;**

**ListInsert\_DuL(L, i, e);**

**}**

**ListTraverse\_DuL(L);**

**printf("请输入需要插入的位置及元素值：\n");**

**scanf("%d %d", &loc1, &e1);**

**ListInsert\_DuL(L, loc1, e1);**

**ListTraverse\_DuL(L);**

**printf("请输入需要删除的位置：\n");**

**scanf("%d", &loc2);**

**ListDelete\_DuL(L, loc2, e2);**

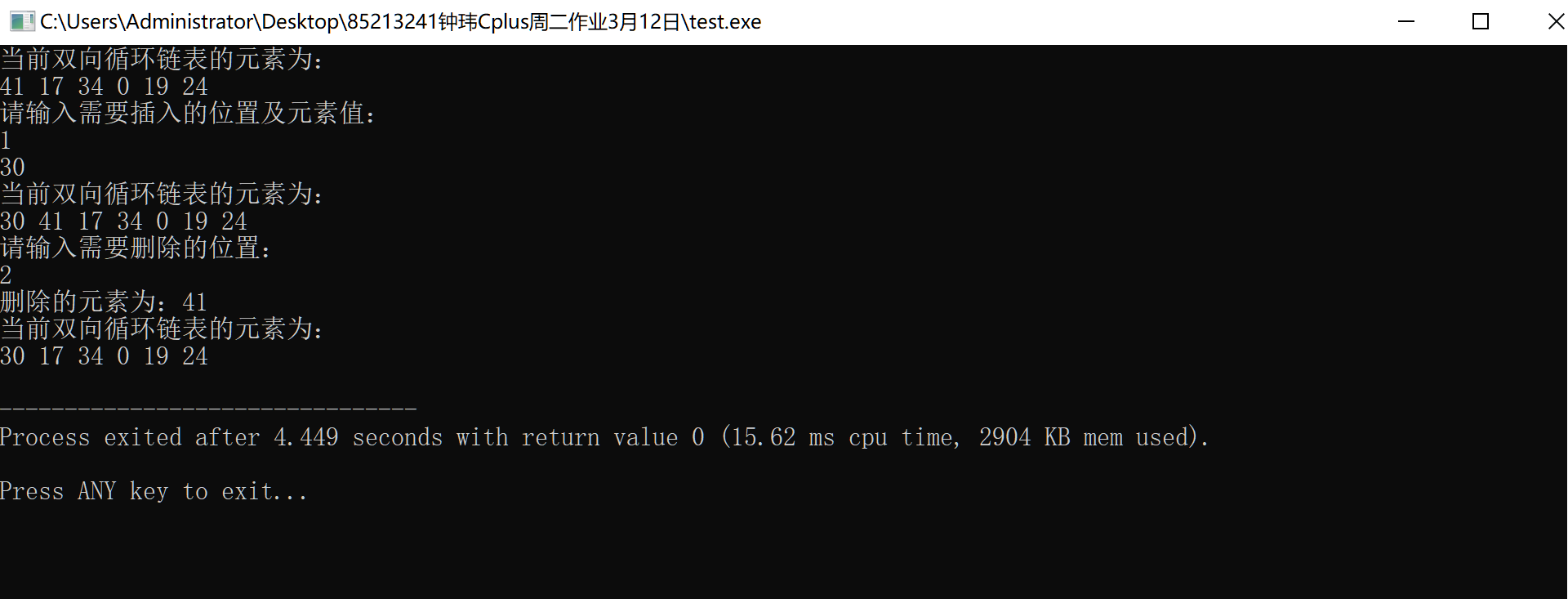
**printf("删除的元素为：%d\n", e2);**

**ListTraverse\_DuL(L);**

**}**

1. **实验结果**

**实验运行结果的截图，如main函数运行结果截图：**

****