** **

**课程实验报告**

课程名： 数据结构实验

学 院： 数学与计算机学院 系 计算机科学与技术系

专 业： 计算机科学与技术

班 级： 计算机222班

对齐

学 号： 5418122020

姓 名： 马星

任课教师： 任 燕

授课学期： 2022 年~~~~ 2023 年 1 学期

**目 录**

[实验一 顺序表](#_Toc31771)

[实验二 XXXXXX](#_Toc32327)

[实验三 XXXXXX](#_Toc2598)

**实验一顺序表**

学生姓名： 马星 学 号： 5418122020 专业班级： 计算机222班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期： 实验成绩：

一、实验项目名称

二、实验目的

三、实验基本原理

四、主要仪器设备及耗材

五、实验步骤

六、实验数据及处理结果

七、思考讨论题或体会或对改进实验的建议

八、参考资料

*#include*<stdio.h>  
*#include*<stdlib.h>  
*#include* <stdbool.h>  
*#include* "LinkedList.h"  
  
  
  
Status init\_LinkedList(LinkedList \*L) {  
 \*L = malloc(*sizeof*(LNode));  
 *if* (!\*L) exit(OVERFLOW);  
 (\*L)->next = NULL;  
 *return* OK;  
}  
  
*int* getLength\_LinkedList(LinkedList L) {  
 LinkedList p = L->next;  
 *int* len = 0;  
 *while* (p) {  
 len++;  
 p = p->next;  
 }  
 *return* len;  
}  
Status append\_LinkedList(LinkedList \*L, *int* e){  
 LinkedList p=(\*L);  
 *while*(p->next){  
 p=p->next;  
 }  
 LNode\* added= malloc(*sizeof* (LNode));  
 added->data=e;  
 added->next=NULL;  
 p->next=added;  
 *return* OK;  
}  
*/\*\*  
 \* 按索引插入元素到链表  
 \* @param L  
 \* @param n 索引  
 \* @param e 插入的元素  
 \* @return  
 \*/*Status insert\_LinkedList(LinkedList \*L, *int* n, *int* e) {  
 LinkedList p = (\*L);  
 *for* (*int* i = 0; i < n; i++){  
 p = p->next;  
 *if* (!p) *return* INFEASIBLE;  
 }  
 LinkedList s = malloc(*sizeof*(LNode));  
 s->data = e;  
 s->next = p->next;  
 p->next = s;  
 *return* OK;  
}  
  
Status getElem\_LinkedList(LinkedList L, *int* n, *int* \*e) {  
 LinkedList p = L->next;  
 *int* i;  
 *for* (i = 1; i < n; i++)  
 p = p->next;  
 *if* (!p || i > n) *return* ERROR;  
  
 \*e = p->data;  
 *return* OK;  
}  
  
*/\*\*  
 \* 初始化一个含n个节点的链表,节点值在控制台输入,节点顺序与输入顺序相反  
 \* @param L  
 \* @param n  
 \*/  
void* create\_LinkedList(LinkedList \*L, *int* n) {  
 (\*L) = malloc(*sizeof*(LNode));  
 *if* (!\*L) exit(OVERFLOW);  
 (\*L)->next = NULL;  
  
 *for* (*int* i = n; i > 0; i--) {  
 LinkedList p = malloc(*sizeof*(LNode));  
 scanf\_s("%d", &p->data);  
 p->next = (\*L)->next;  
 (\*L)->next = p;  
 }  
}  
  
*void* clear\_LinkedList(LinkedList \*L) {  
 LinkedList p = (\*L)->next;  
 *while* (p != NULL) {  
 LinkedList q = p;  
 p = p->next;  
 free(q);  
 }  
 (\*L)->next = NULL;  
}  
  
bool isEmpty\_LinkedList(LinkedList L) {  
 *return* !L->next;  
}  
  
*int* locateElem\_LinkedList(LinkedList L, *int* e, bool(\*compare)(*int*, *int*)) {  
 LinkedList p = L->next;  
 *int* i = 1;  
 *while* (p) {  
 *if* (compare(e, p->data))  
 *return* i;  
 i++;  
 p = p->next;  
 }  
 *return* ERROR;  
}  
  
*/\*\*  
 \* 获取前驱  
 \* @param L  
 \* @param cur\_e  
 \* @param pre\_e  
 \* @return  
 \*/*Status priorElem\_LinkedList(LinkedList L, *int* cur\_e, *int* \*pre\_e) {  
 LinkedList p = L->next;  
 LinkedList q = L;  
 *while* (p && p->data != cur\_e) {  
 q = q->next;  
 p = p->next;  
 }  
 *if* (!p) exit(ERROR);  
 \*pre\_e = q->data;  
 *return* OK;  
}  
*/\*\*  
 \* 获取后继  
 \* @param L  
 \* @param cur\_e  
 \* @param nex\_e  
 \* @return  
 \*/*Status nextElem\_LinkedList(LinkedList L, *int* cur\_e, *int* \*nex\_e) {  
 LinkedList p = L;  
 *while* (p->next && p->data != cur\_e)  
 p = p->next;  
 *if* (!p->next) exit(ERROR);  
 \*nex\_e = p->next->data;  
 *return* OK;  
}  
*/\*\*  
 \* 插入节点到链表元素值为e的节点前  
 \* @param L  
 \* @param e  
 \* @param insertVal  
 \* @return  
 \*/*Status insertByVal\_LinkedList(LinkedList \*L, *int* e, *int* insertVal) {  
 LinkedList p = (\*L)->next;  
 LinkedList prev = \*L;  
 *while* (p && p->data != e) {  
 p = p->next;  
 prev = prev->next;  
 }  
 *if* (!p) exit(ERROR);  
 LinkedList s = malloc(*sizeof*(LNode));  
 s->data = insertVal;  
 s->next = p;  
 prev->next = s;  
 *return* OK;  
}  
*/\*\*  
 \* 按值删除元素  
 \* @param L  
 \* @param e 要删除的元素值  
 \* @return  
 \*/*Status delete\_LinkedList(LinkedList \*L, *int* e) {  
 LinkedList p = (\*L)->next;  
 LinkedList prev = \*L;  
  
 *while* (p && p->data != e) {*//找到删除节点以及它的前驱* p = p->next;  
 prev = prev->next;  
 }  
 *if* (!p) exit(ERROR);  
 prev->next = p->next;  
 free(p);  
 *return* OK;  
}  
*/\*\*  
 \* 合并升序链表Lb到升序链表La  
 \* @param La  
 \* @param Lb  
 \* @return  
 \*/*Status merge\_LinkedList(LinkedList \*La, LinkedList \*Lb) { *//用于非递减链表* LinkedList p = \*La;  
 LinkedList Pa = (\*La)->next;  
 LinkedList Pb = (\*Lb)->next;  
 *while* (Pa && Pb) {  
 *if* (Pa->data > Pb->data) {  
 p->next = Pb;  
 p = Pb;  
 Pb = Pb->next;  
 } *else* {  
 p->next = Pa;  
 p = Pa;  
 Pa = Pa->next;  
 }  
 }  
 p->next = Pa ? Pa : Pb; *// 插入剩余段* free(Lb); *//释放Lb的头节点  
 return* OK;  
}  
*/\*\*  
 \* 链表反转  
 \* @param L  
 \* @return  
 \*/*Status reverse\_LinkedList(LinkedList \*L) {  
 LinkedList curr = (\*L)->next;  
 LinkedList prev = NULL, next = NULL;  
 *while* (curr) {  
 next = curr->next;  
 curr->next = prev;  
 prev = curr;  
 curr = next;  
 }  
 (\*L)->next = prev;  
 *return* OK;  
}  
  
*void* printLinkedList(LinkedList L) {  
 LinkedList p = L->next;  
 *while* (p != NULL) {  
 printf("%d->", p->data);  
 p = p->next;  
 }  
 printf("null\n");  
}

*每个实验以新页开始，阅读后删除*

**实验二 XXXXXX**

学生姓名： 马星 学 号： 5418122020 专业班级： 计算机222班 实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期： 实验成绩：

一、实验项目名称

二、实验目的

三、实验基本原理

四、主要仪器设备及耗材

五、实验步骤（完整内容见光盘）

六、实验数据及处理结果

七、思考讨论题或体会或对改进实验的建议

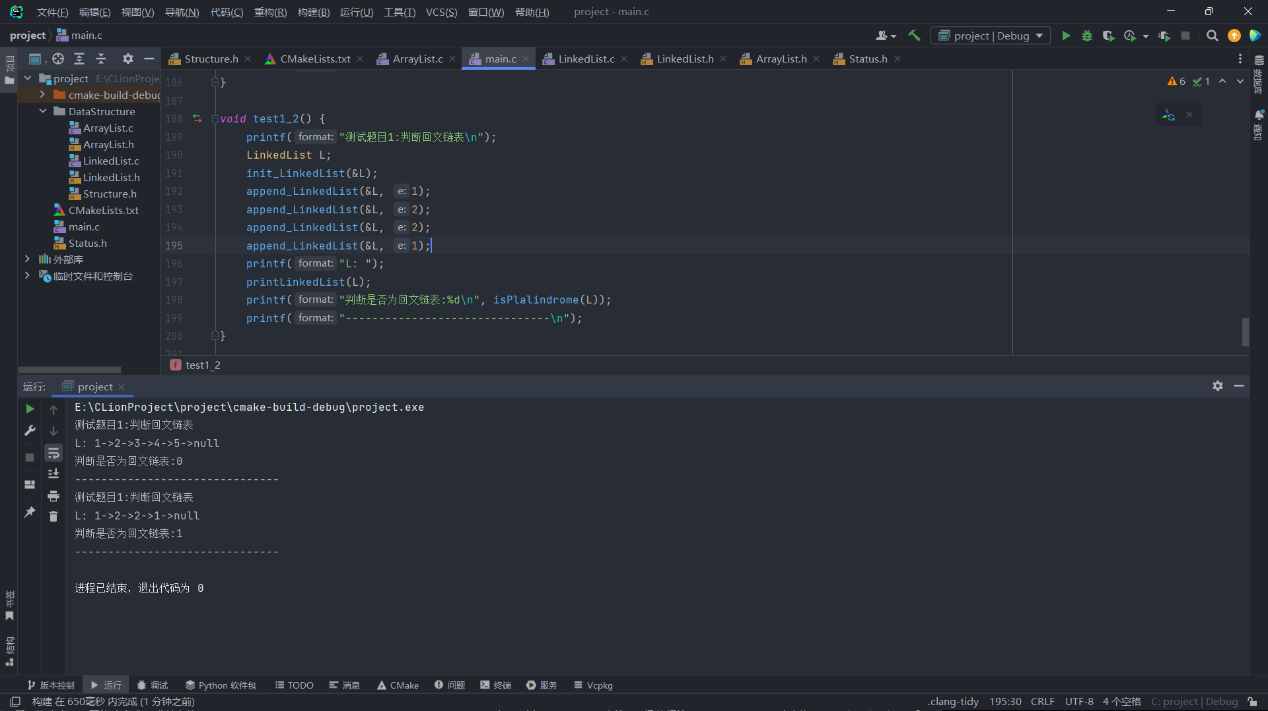
八、参考资料

*//  
// Created by 绝迹的星 on 2023/9/26.  
//  
  
#include* <stdio.h>  
*#include* <stdbool.h>  
*#include* "Status.h"  
*#include* "DataStructure/LinkedList.h"  
  
*void* test1\_1();  
  
*void* test1\_2();  
  
*void* test2\_1();  
  
*void* test2\_2();  
  
*void* test2\_3();  
*/\*\*  
 \* <h1>判断回文链表</h1>  
 \* 快慢指针法反转前半部分链表  
 \* 然后从前半部分链表的新头节点与后半部分链表头开始依次比较  
 \* @return 是回文链表返回true(1),不是则返回false(0)  
 \*/*bool isPlalindrome(LinkedList L) {  
 LNode \*p = NULL;  
 LNode \*slow = L->next;*//快慢指针* LNode \*fast = L->next;  
 *while* (fast) {  
 fast = fast->next;  
 *if* (!fast) {  
 slow = slow->next;  
 *break*;  
 }  
 fast = fast->next;  
 *//链表反转* LNode \*temp = slow->next;  
 slow->next = p;  
 p = slow;  
  
 slow = temp;  
 }  
 *while* (slow && p) {  
 *if* (slow->data != p->data) {  
 *return* false;  
 }  
 slow = slow->next;  
 p = p->next;  
 }  
 *return* true;  
}

测试代码:

*void* test1\_1() {  
 printf("测试题目1:判断回文链表\n");  
 LinkedList L;  
 init\_LinkedList(&L);  
 append\_LinkedList(&L, 1);  
 append\_LinkedList(&L, 2);  
 append\_LinkedList(&L, 3);  
 append\_LinkedList(&L, 4);  
 append\_LinkedList(&L, 5);  
 printf("L: ");  
 printLinkedList(L);  
 printf("判断是否为回文链表:%d\n", isPlalindrome(L));  
 printf("-------------------------------\n");  
}  
  
*void* test1\_2() {  
 printf("测试题目1:判断回文链表\n");  
 LinkedList L;  
 init\_LinkedList(&L);  
 append\_LinkedList(&L, 1);  
 append\_LinkedList(&L, 2);  
 append\_LinkedList(&L, 2);  
 append\_LinkedList(&L, 1);  
 printf("L: ");  
 printLinkedList(L);  
 printf("判断是否为回文链表:%d\n", isPlalindrome(L));  
 printf("-------------------------------\n");  
}

测试结果:



*每个实验以新页开始，阅读后删除*

**实验三 XXXXXX**

学生姓名： 马星 学 号： 5418122020 专业班级： 计算机222班

实验类型：□ 验证 □ 综合 □ 设计 □ 创新 实验日期： 实验成绩：

一、实验项目名称

二、实验目的

三、实验基本原理

四、主要仪器设备及耗材

五、实验步骤（完整内容见光盘）

六、实验数据及处理结果

七、思考讨论题或体会或对改进实验的建议

八、参考资料

*/\*\*  
 \* <h1>分隔链表:实现1</h1>  
 \* 头插法  
 \*/  
void* split(LinkedList \*L, *int* x) {  
 LNode \*prev = \*L;  
 LNode \*p = (\*L)->next;  
 *while* (p) {  
 *if* (p->data < x) {  
 *//将节点插入到表头* LNode \*next = p->next;  
 prev->next = p->next;  
 p->next = (\*L)->next;  
 (\*L)->next = p;  
 p = next;  
 *if* (prev == \*L) {*//前驱为头节点需要特殊处理* prev = prev->next;  
 }  
 } *else* {  
 prev = p;  
 p = p->next;  
 }  
 }  
}  
  
*/\*\*  
 \*<h1>分隔链表:实现2</h1>  
 \* 交换值法  
 \*/  
void* split2(LinkedList \*L, *int* x) {  
 *if* (!(\*L)->next) {  
 *return*;  
 }  
 LNode \*p1 = (\*L)->next;  
 LNode \*p2 = (\*L)->next;  
 *while* (p1 && p2) {  
 *//p2寻找大于x的节点  
 while* (p2 && p2->data < x) {  
 p2 = p2->next;  
 }  
 *//p1从第一个大节点开始,寻找小于x的节点* p1 = p2;  
 *while* (p1 && p1->data >= x) {  
 p1 = p1->next;  
 }  
 *if* (p1 && p2) {*//交换节点值  
 int* temp = p1->data;  
 p1->data = p2->data;  
 p2->data = temp;  
 }  
 }  
}

测试代码:

*void* test2\_1() {  
 printf("测试题目2:链表分隔\n");  
 LinkedList L;  
 init\_LinkedList(&L);  
 append\_LinkedList(&L, 1);  
 append\_LinkedList(&L, 8);  
 append\_LinkedList(&L, 5);  
 append\_LinkedList(&L, 2);  
 append\_LinkedList(&L, 3);  
 append\_LinkedList(&L, 7);  
 append\_LinkedList(&L, 4);  
 append\_LinkedList(&L, 6);  
 printf("L: ");  
 printLinkedList(L);  
  
 split(&L, 7);  
 printf("以x=7分隔链表L,结果为:");  
 printLinkedList(L);  
 printf("-------------------------------\n");  
}  
  
*void* test2\_2() {  
 printf("测试题目2:链表分隔\n");  
 LinkedList L;  
 init\_LinkedList(&L);  
 append\_LinkedList(&L, 1);  
 append\_LinkedList(&L, 4);  
 append\_LinkedList(&L, 3);  
 append\_LinkedList(&L, 2);  
 append\_LinkedList(&L, 5);  
 append\_LinkedList(&L, 2);  
 printf("L: ");  
 printLinkedList(L);  
  
 split(&L, 3);  
 printf("以x=3分隔链表,结果为:");  
 printLinkedList(L);  
 printf("-------------------------------\n");  
}  
  
*void* test2\_3() {  
 printf("测试题目2:链表分隔\n");  
 LinkedList L;  
 init\_LinkedList(&L);  
 append\_LinkedList(&L, 1);  
 append\_LinkedList(&L, 4);  
 append\_LinkedList(&L, 3);  
 append\_LinkedList(&L, 2);  
 append\_LinkedList(&L, 5);  
 append\_LinkedList(&L, 2);  
 printf("L: ");  
 printLinkedList(L);  
  
 split2(&L, 3);  
 printf("(实现2)以x=3分隔链表,结果为:");  
 printLinkedList(L);  
 printf("-------------------------------\n");  
}

测试结果:

