实验2单链表实验

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 姓名 | 张渊 | 学号 | 2022210000 | 专业班级 | 计科22-0 |
| 指导教师 | 张先宜 | 实验时间 | 2023.4.5 | 实验地点 |  |

**实验名称：实验二 单链表实验**

**2.1实验目的**

1. 熟练掌握线性表的链式存储结构。
2. 熟练掌握单链表的有关算法设计和实现。
3. 根据具体问题的需要，设计出合理的表示数据的链式存储结构，并设计相关算法。

**2.2 实验要求**

1. 本次实验中的链表结构指带头结点的单链表；
2. 单链表结构和运算定义，算法的实现以库文件方式实现，不得在测试主程序中直接实现；比如存储、算法实现放入文件：linkedList.h
3. 程序运行、测试正确；
4. 实验程序有较好可读性，各运算和变量的命名直观易懂，符合软件工程要求；
5. 程序有适当的注释，程序的书写要采用缩进格式。

**2.3 实验任务**

编写算法实现下列问题的求解。

1. 在第i个结点位置插入值为x的结点。

实验测试数据基本要求：

第一组数据：单链表长度n≥10，x=100, i分别为5,n,n+1,0,1,n+2

第二组数据：单链表长度n=0，x=100，i=5

1. 删除单链表中第i个元素结点。

实验测试数据基本要求：

第一组数据：单链表长度n≥10，i分别为5,n,1,n+1,0

第二组数据：单链表长度n=0， i=5

1. 在一个递增有序的单链表L中插入一个值为x的元素，并保持其递增有序特性。

实验测试数据基本要求：

单链表元素为 （10,20,30,40,50,60,70,80,90,100）,

x分别为25，85，110和8

1. 将单链表Ｌ中的奇数项和偶数项结点分解开（元素值为奇数、偶数），分别放入新的单链表中，然后原表和新表元素同时输出到屏幕上，以便对照求解结果。

实验测试数据基本要求：

第一组数据：单链表元素为 （1,2,3,4,5,6,7,8,9,10,20,30,40,50,60）

第二组数据：单链表元素为 （10,20,30,40,50,60,70,80,90,100）

1. 求两个递增有序单链表L1和L2中的公共元素，放入新的单链表L3中。

实验测试数据基本要求：

第一组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10，18，20，30）

第二组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

第二个单链表元素为 （2，4，5，7，8，9，12，22）

第三组

第一个单链表元素为 （）

第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10）

1. 删除递增有序单链表中的重复元素，要求时间性能最好。

实验测试数据基本要求：

第一组数据：单链表元素为 （1,2,3,4,5,6,7,8,9）

第二组数据：单链表元素为 （1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,9）

第三组数据：单链表元素为 （1,2,3,4,5,5,6,7,8,8,9,9,9,9,9）

1. 递增有序单链表L1、L2，不申请新结点，利用原表结点对两表进行合并，并使得合并后成为一个集合，合并后用L1的头结点作为头结点，删除多余的结点，删除L2的头结点。要求时间性能最好。

实验测试数据基本要求：

第一组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10，18，20，30）

第二组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

第二个单链表元素为 （2，4，5，7，8，9，12，22）

第三组

第一个单链表元素为 （）

第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10）

**2.4\* 链表扩展实验**

（非必做内容，有兴趣的同学选做）

1. （递增有序）单链表表示集合A、B，实现：

C=A∪B，C=A∩B，C=A-B

A=A∪B，A=A∩B，A=A-B

1. (2009)（15分）已知一个带有表头结点的单链表，结点结构如下图。假设该链表只给出了头指针list。在不改变链表的前提下，请设计一个尽可能高效的算法，查找链表中倒数第k个位置上的结点（k为正整数）。若查找成功，算法输出该结点的data值，并返回1；否则，只返回0。要求：

（1）描述算法的基本设计思想

（2）描述算法的详细实现步骤

（3）根据设计思想和实现步骤，采用程序设计语言描述算法（使用C 或C++语言实现），关键之处请给出简要注释。

**2.4 数据结构设计**

typedef struct LNode

{

int data;

struct LNode\* next;

}LNode, \* LinkList;//前者侧重表示节点 后者侧重表示链表

**2.5算法设计**

**1、2**均为书上的基本运算，故算法设计思想省略。

3【算法思想】

遍历单链表，保证所指的结点的值<x同时结点p的下一个结点不为空。遍历结束后在此时p结点后插入新的节点即可

【算法描述】

void InsertSort(LinkList& L, int x)

{

LNode\* s, \* p = L;

while (p->next != NULL && p->next->data < x)

p = p->next;

InsertNextNode(p, x);

}

4【算法思想】

建立新的单链表L1,L2分别存储奇数偶数数据，遍历L，将其中的数据分别存入L1,L2即可

【算法描述】

//分为奇偶单链表

void part(LinkList L, LinkList &L1, LinkList &L2)

{

LNode\* s, \* p = L->next;

LNode\* p1 = L1, \* p2 = L2;

while (p != NULL)

{

if (p->data % 2 == 0)//为偶数，进入L1中

{

s = (LNode\*)malloc(sizeof(LNode));

s->data = p->data;

s->next = p1->next;

p1->next = s;

p1 = s;

p = p->next;

}

else//为奇数，进入L2中

{

s = (LNode\*)malloc(sizeof(LNode));

s->data = p->data;

s->next = p2->next;

p2->next = s;

p2 = s;

p = p->next;

}

}

}

5【算法思想】

通过双指针遍历L1、L2，L1、L2的数据相同就存入L3并移动指针，在L1和L2对应数据不相同时移动对应的指针直到L1和L2有一个指向空结束

【算法描述】

//求单链表L1,L2的公共元素存入L3

void listCommon(LinkList L1, LinkList L2, LinkList& L3)

{

LNode\* s, \* p1, \* p2, \* p3 = L3;

p1 = L1->next;

p2 = L2->next;

while (p1 != NULL && p2 != NULL)

{

if (p1->data == p2->data)

{

s = (LNode\*)malloc(sizeof(LNode));

s->data = p1->data;

s->next = NULL;

p3->next = s;

p3 = s;

p1 = p1->next;

p2 = p2->next;

}

else if (p1->data > p2->data)

p2 = p2->next;

else

p1 = p1->next;

}

}

6【算法思想】

建立两个指针，一个指向p的下一个结点，另一个指向p的下下一个结点，使P不断后移直到p指向NULL,如果p和q指向的数据相同就让p指向q并移动q，反之移动p和q即可

【算法描述】

//删除递增有序单链表中的重复元素

void incListDelete(LinkList& L)

{

LNode\* p, \* q;

p = L->next;

q = L->next->next;

while (q != NULL)

{

if (p->data == q->data)

{

q = q->next;

p->next = q;

}

else

{

p = p->next;

q = q->next;

}

}

}

7【算法思想】

(1)先声明两个指针p、q分别指向单链表的头结点L1、L2

(2)不断使指针p后移直到p指向NULL,最终让p指向q指针即可

【算法描述】

//合并单链表L1,L2至L1

void ListUnite(LinkList& L1, LinkList& L2)

{

LNode\* p = L1, \* q = L2;

while (p->next != NULL)

p = p->next;

p->next = q->next;

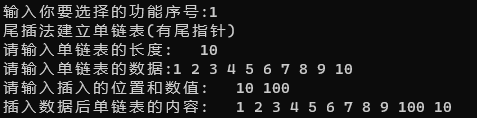
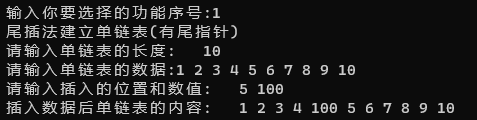
free(L2);

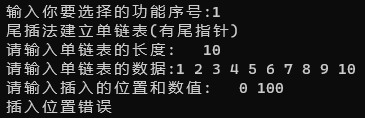
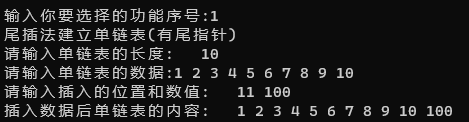
}

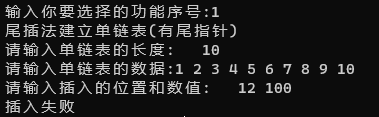
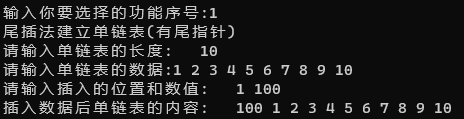
**2.6运行和测试**

**1**

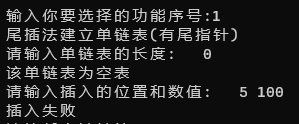
第一组数据：单链表长度n≥10，x=100, i分别为5,n,n+1,0,1,n+2





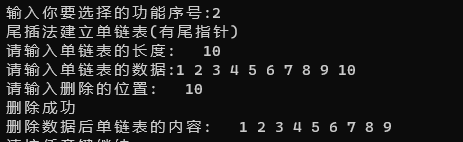
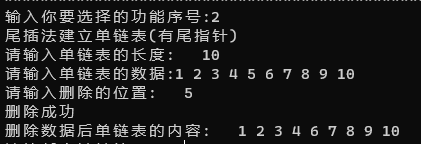


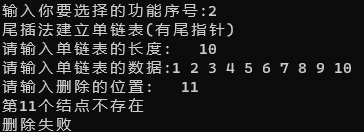
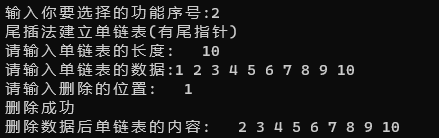
第二组数据：单链表长度n=0，x=100，i=5

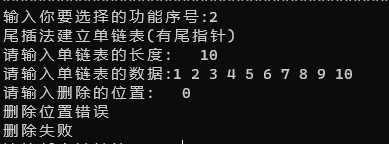


**2**

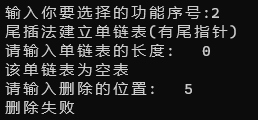
第一组数据：单链表长度n≥10，i分别为5,n,1,n+1,0







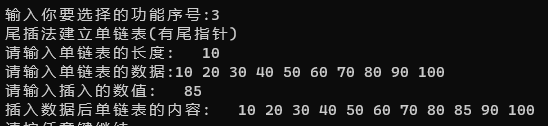
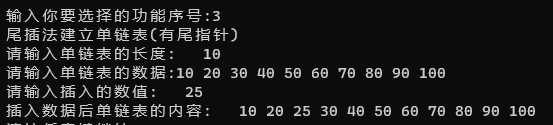
第二组数据：单链表长度n=0， i=5

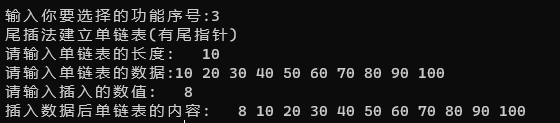
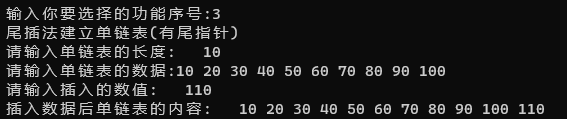


**3**

单链表元素为 （10,20,30,40,50,60,70,80,90,100）,

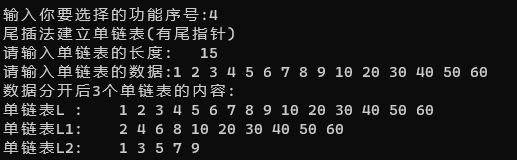
x分别为25，85，110和8



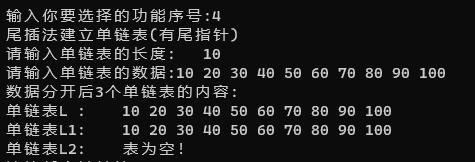


**4**

第一组数据：单链表元素为 （1,2,3,4,5,6,7,8,9,10,20,30,40,50,60）



第二组数据：单链表元素为 （10,20,30,40,50,60,70,80,90,100）

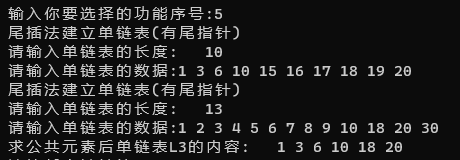


**5**

第一组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

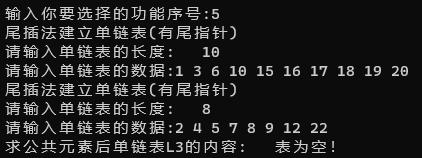
第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10，18，20，30）



第二组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

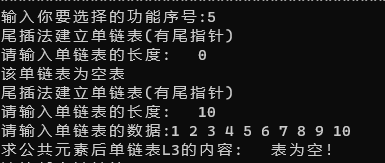
第二个单链表元素为 （2，4，5，7，8，9，12，22）



第三组

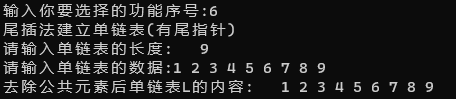
第一个单链表元素为 （）

第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10）

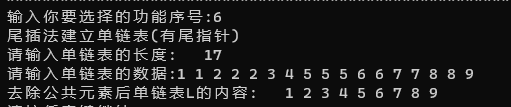


**6**

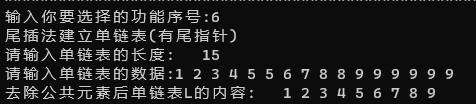
第一组数据：单链表元素为 （1,2,3,4,5,6,7,8,9）



第二组数据：单链表元素为 （1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,9）



第三组数据：单链表元素为 （1,2,3,4,5,5,6,7,8,8,9,9,9,9,9）

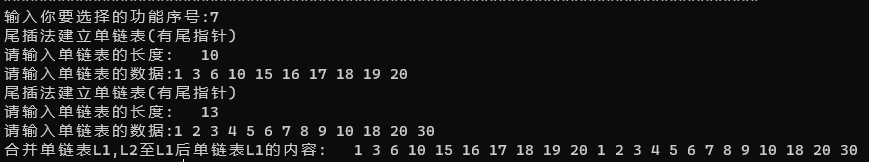


**7**

第一组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

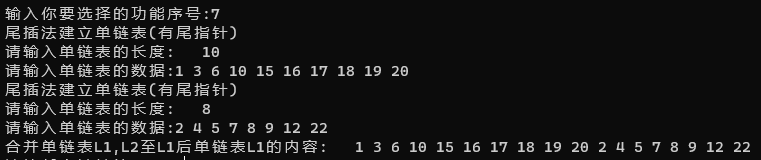
第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10，18，20，30）



第二组

第一个单链表元素为 （1，3，6，10，15，16，17，18，19，20）

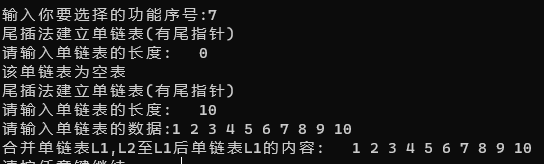
第二个单链表元素为 （2，4，5，7，8，9，12，22）



第三组

第一个单链表元素为 （）

第二个单链表元素为 （1，2，3，4，5，6，7，8，9，10）



**2.7总结、心得和建议**

本次实验在结构设计时分别建立了LNode和 \* LinkList分别侧重强调节点和链表，方便了后续查看代码。对指针传递和引用传递有了更深刻的理解，指针传递的本质其实还是值传递，只不过保存了传入变量的地址，只有引用传递是对原来变量的修改。此外，函数的初始化也十分重要，在以后建立变量时尽量都进行初始化。在调试代码时函数的返回值多次出现问题，应当根据需要确立适当的函数返回值。对函数的调用也应当认真思考，有许多步骤可以省略提取简化。

此次实验内容和实验一顺序表的实验内容大体相同，应当学会灵活变态。本次实验其实耗时也挺长的，希望自己继续努力，仔细、认真完成每一次实验，并在实验过程中发现平时学习的疏漏与不足，不断提高自己的能力。

**[2.8 附录]**

（源代码清单。纸质报告不做要求。电子版报告，可直接附源文件，删除编译生成的所有文件）

****