数据结构实验报告

实验成绩：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 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：实验一 顺序表实验**

**1.1实验目的**

1. 熟练掌握线性表的顺序存储结构。
2. 熟练掌握顺序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元素的顺序结构，并设计相关算法。

**1.2实验要求**

1. 顺序表结构和运算定义，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：seqList.h
2. 程序有适当的注释，程序的书写要采用缩进格式。
3. 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；
4. 程序要具有一定的健壮性，即当输入数据非法时，程序也能适当地做出反应，如插入删除时指定的位置不对等等。
5. 程序要做到界面友好，在程序运行时用户可以根据相应的提示信息进行操作。
6. 程序运行、测试正确；
7. 根据实验报告模板详细书写实验报告。

**1.3 实验内容**

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：顺序表长度n≥10，x=100, i分别为5,n,n+1,0,1,n+2

第二组数据：顺序表长度n=0，x=100，i=5

1. 删除顺序表中第i个元素结点。

实验测试数据基本要求：

第一组数据：顺序表长度n≥10，i分别为5,n,1,n+1,0

第二组数据：顺序表长度n=0， i=5

1. 在一个递增有序的顺序表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：

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

x分别为25，85，110和8

1. 将顺序表Ｌ中的奇数项和偶数项结点分解开（元素值为奇数、偶数），分别放入新的顺序表中，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：顺序表元素为 （1,2,3,4,5,6,7,8,9,10,20,30,40,50,60）

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

1. 求两个递增有序顺序表L1和L2中的公共元素，放入新的顺序表L3中。

实验测试数据基本要求：

第一组

第一个顺序表元素为 （1，3，6，10，15，16，17，18，19，20）

第二个顺序表元素为 （1，2，3，4，5，6，7，8，9，10，18，20，30）

第二组

第一个顺序表元素为 （1，3，6，10，15，16，17，18，19，20）

第二个顺序表元素为 （2，4，5，7，8，9，12，22）

第三组

第一个顺序表元素为 （）

第二个顺序表元素为 （1，2，3，4，5，6，7，8，9，10）

1. 删除递增有序顺序表中的重复元素，并统计移动元素次数，要求时间性能最好。

实验测试数据基本要求：

第一组数据：顺序表元素为 （1,2,3,4,5,6,7,8,9）

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

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

//\* 顺序表扩展实验

（非必做内容，有兴趣的同学选做）

1.（递增有序）顺序表表示集合A、B，实现：

C=A∪B，C=A∩B，C=A-B

A=A∪B，A=A∩B，A=A-B

2.（递增有序）顺序表表示集合A、B，判定A是否B的子集。

3. (2011)（15 分）一个长度为L（L≥1）的升序序列S，处在第 个位置的数称为S 的中位数。例如，若序列S1=(11, 13, 15, 17, 19)，则S1 的中位数是15。两个序列的中位数是含它们所有元素的升序序列的中位数。例如，若S2=(2, 4, 6, 8, 20)，则S1 和S2 的中位数是11。



现有两个等长升序序列A 和B，试设计一个在时间和空间两方面都尽可能高效的算法，找出两个序列A 和B 的中位数。要求：

（1）给出算法的基本设计思想。

（2）根据设计思想，采用C 或C++语言描述算法，关键之处给出注释。

（3）说明你所设计算法的时间复杂度和空间复杂度。

**1.4 数据结构设计**

typedef struct seqList

{

int data[MaxSize]; //存储表中元素的数组

int listLen; //表长度分量

}seqList;

**1.5 算法设计**

1与2均为书上的基本运算，故算法设计思想省略。

3【算法思想】

将所要插入的数先添加到线性表的末尾 然后逐个与前面的数进行比较 若比前面的数小，就交换两数 直到找到<=该数的数即可

【算法描述】

//在一个递增有序的顺序表L中插入一个值为x的元素，并保持其递增有序特性

bool listInsert2(seqList& L, int x)

{

if (L.listLen + 1 > MaxSize)

{

cout << "表已满，插入失败" << endl;

return false;

}

L.listLen++;

L.data[L.listLen - 1] = x; //把插入的数放在顺序表的末尾

for (int i = L.listLen - 1; i > 0; i--)

{

if (L.data[i] < L.data[i - 1]) //插入的数比相邻之前的数小，则互换位置

{

int temp = L.data[i];

L.data[i] = L.data[i - 1];

L.data[i - 1] = temp;

}

else //如果插入的数比之前的都大，则满足递增要求退出

return true;

}

return true;

}

4【算法思想】

（1）先创建2个线性表，分别存放奇数和偶数

（2）之后将原来线性表的数据分别放入创建的2个线性表

【算法描述】

//顺序表中奇数项和偶数项分开

bool listSeperate(seqList L, seqList& L1, seqList& L2)

{

int a = 0; //记录L1表的长度 L1为偶数表

int b = 0; //记录L2表的长度 L2为奇数表

for (int i = 0; i < L.listLen; i++)

{

if (L.data[i] % 2 == 0)

L1.data[a++] = L.data[i];

else

L2.data[b++] = L.data[i];

}

L1.listLen = a - 1;//不要忘了表长-1

L2.listLen = b - 1;

return true;

}

5【算法思想】

(1)通过双指针，分别对顺序表L1 L2进行遍历

(2)判断L1 L2的对应数据的大小,并不断移动 将符合要求的数据存入线性表L3

【算法描述】

//求两个递增有序顺序表L1和L2中的公共元素

bool listBoth(seqList L1, seqList L2, seqList& L3)

{

int i = 0, j = 0, k = 0;

while (i < L1.listLen && j < L2.listLen) //减少遍历次数，只要有一个表遍历结束则结束

{

if (L1.data[i] == L2.data[j])

{

L3.data[k] = L1.data[i];

i++; j++; k++;

}

else if (L1.data[i] > L2.data[j])

j++;

else

i++;

}

L3.listLen = k;

return true;

}

6【算法思想】

（1）从后往前删除判断相邻的两个数是否相同（同时不断前移p 代表最后一个数的下标）

（2）由于删除过程表长会改变，应当先保存初始表长

【算法描述】

//删除递增有序顺序表中的重复元素，并统计移动元素次数

bool listCancle(seqList& L)

{

int count = 0; //统计移动元素次数

int p = L.listLen - 1;

int size = L.listLen; //由于删除过程中L->listLen的值会发生改变，所以先把顺序表的初始长度记录下来

for (int i = 1; i < size; i++)

{

if (L.data[p] == L.data[p - 1])

{

listDelete(L, p + 1);

count += L.listLen - p;

}

p--; //跳到前一个元素

}

cout << "删除重复元素以后的新表为：";

listOutput(L);

cout << "移动元素的次数为：" << count << endl;

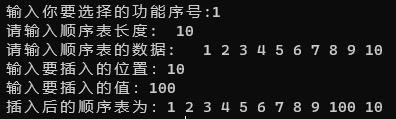
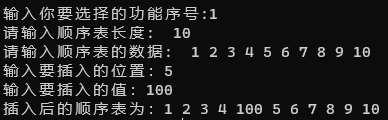
return true;

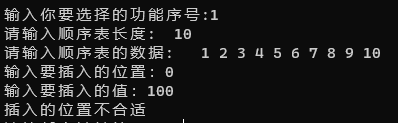
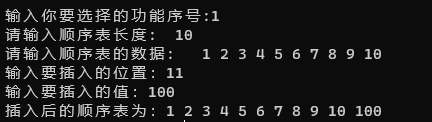
}

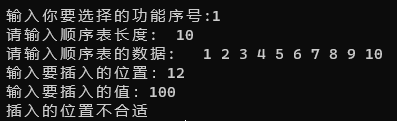
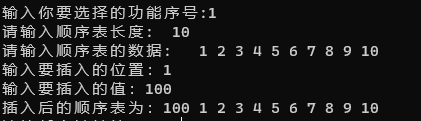
**1.6 运行和测试**

1.

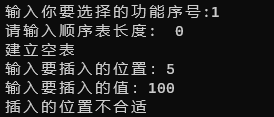
第一组数据：顺序表长度n≥10，x=100, i分别为5,n,n+1,0,1,n+2（n=10）





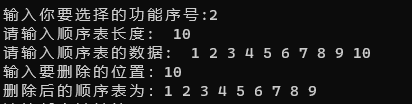
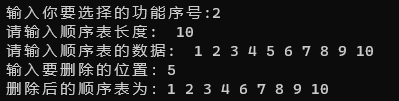


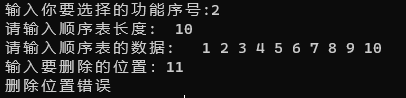
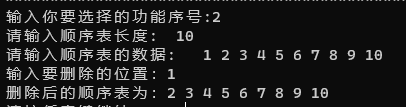
第二组数据：顺序表长度n=0，x=100，i=5

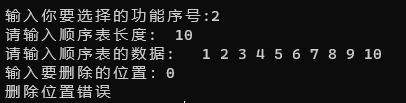


**2.**

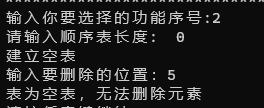
第一组数据：顺序表长度n≥10，i分别为5,n,1,n+1,0 (n=10)







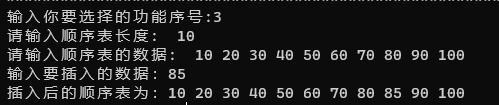
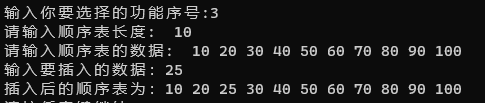
第二组数据：顺序表长度n=0， i=5

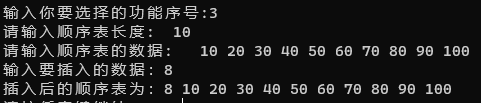
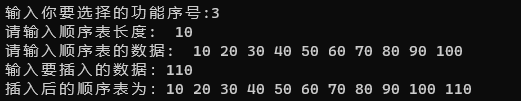


**3.**

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

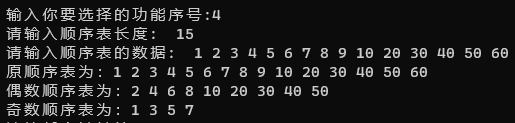
x分别为25，85，110和8



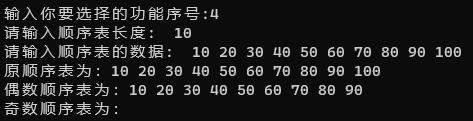


**4.**

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



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

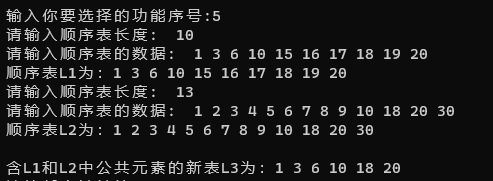


5.

第一组

第一个顺序表元素为 （1，3，6，10，15，16，17，18，19，20）

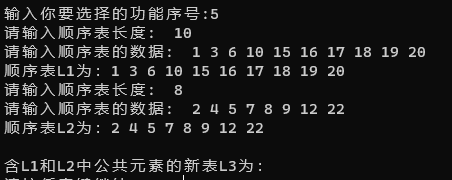
第二个顺序表元素为 （1，2，3，4，5，6，7，8，9，10，18，20，30）



第二组

第一个顺序表元素为 （1，3，6，10，15，16，17，18，19，20）

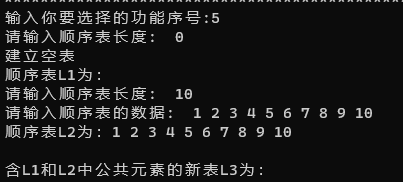
第二个顺序表元素为 （2，4，5，7，8，9，12，22）



第三组

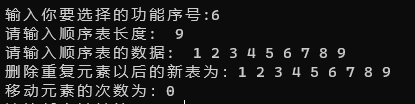
第一个顺序表元素为 （）

第二个顺序表元素为 （1，2，3，4，5，6，7，8，9，10）

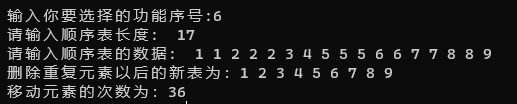


6.

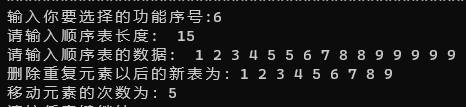
第一组数据：顺序表元素为 （1,2,3,4,5,6,7,8,9）



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



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



**1.7 总结、心得和建议**

本次实验是第一次实验，暴露出许多问题。在线性表的建立和函数参数的选择时我没有分清楚值传递、指针传递和引用传递的区别，在写代码运行时多次报错，我也去查了相关资料明白了三者的区别。所以我没有用指针建立线性表，只是建立了对应的对象，在需要对内容修改时进行了引用传递（其实也可以进行指针传递）。在编写去重和排序时从后往前大大提高了效率，学到了新知识。

通过不断地修改调试自己对编程有了更深的理解，增强了独立思考的能录，也学到了新知识。将来要不断增强自己写代码的能力，更上一层楼。

**[1.8 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）

****