

**课 程 实 验 报 告**

**课程名称： 数据结构实验**

**专业班级： 计算机科学与技术**

**学 号：**

**姓 名：**

**指导教师：**

**报告日期： 2021年 5月 7 日**

**计算机科学与技术学院**

目 录

[1 基于顺序存储结构的线性表实现 3](#_Toc458159879)

[1.1 问题描述 3](#_Toc458159880)

[1.2 系统设计 3](#_Toc458159882)

[1.3 系统实现 6](#_Toc458159883)

[1.4 系统测试 1](#_Toc458159884)1

[1.5 实验小结 1](#_Toc458159884)4

[2 基于链式存储结构的线性表实现 1](#_Toc458159885)5

[2.1 问题描述 1](#_Toc458159886)5

[2.2 系统设计 1](#_Toc458159887)5

[2.3 系统实现 1](#_Toc458159888)7

[2.4 系统测试 2](#_Toc458159889)1

[2.5 实验小结 2](#_Toc458159889)4

[3 基于二叉链表的二叉树实现 2](#_Toc458159890)

[3.1 问题描述 2](#_Toc458159891)

[3.2 系统设计 2](#_Toc458159892)

[3.3 系统实现 2](#_Toc458159893)

[3.4 实验小结 2](#_Toc458159894)

[4 基于二叉链表的二叉树实现 2](#_Toc458159895)

[4.1 问题描述 2](#_Toc458159896)

[4.2 系统设计 2](#_Toc458159897)

[4.3 系统实现 2](#_Toc458159898)

[4.4 实验小结 2](#_Toc458159899)

[参考文献 2](#_Toc458159900)

[附录A 基于顺序存储结构线性表实现的源程序 2](#_Toc458159901)5

[附录B 基于链式存储结构线性表实现的源程序 4](#_Toc458159902)4

[附录C 基于二叉链表二叉树实现的源程序 2](#_Toc458159903)

[附录D 基于邻接表图实现的源程序 2](#_Toc458159904)

**1 基于顺序存储结构的线性表实现**

**1.1问题描述**

通过实验达到

⑴加深对线性表的概念、基本运算的理解；

⑵熟练掌握线性表的逻辑结构与物理结构的关系；

⑶物理结构采用顺序表,熟练掌握线性表的基本运算的实现。

**1.2系统设计**

**1.2.1整体结构设计**

实现顺序储存的线性表的基本运算操作。通过switch语句根据菜单系统上的提示内容输入数字进行对应线性表操作，包括：创建，销毁，清空，判空，求表长，获取元素，查找元素，获得前驱，获得后继，插入元素，删除元素，遍历线性表，数据读写文件，多线性表管理（增加一个新线性表，移除一个线性表，查找线性表）及相关附加功能：排序、判断子表、求两表的交集、并集、差集。

**1.2.2数据结构设计**

函数设计：

**⑴初始化表：函数名称是InitList(L)；**

如果线性表L不存在，构造一个空的线性表，返回OK，否则返回INFEASIBLE。

**⑵销毁表：函数名称是DestroyList(L)**；

如果线性表L存在，销毁线性表L，释放数据元素的空间，返回OK，否则返回INFEASIBLE。

**⑶清空表：函数名称是ClearList(L)；**

如果线性表L存在，删除线性表L中的所有元素，返回OK，否则返回INFEASIBLE

**⑷判定空表：函数名称是ListEmpty(L)；**

如果线性表L存在，判断线性表L是否为空，空就返回TRUE，否则返回FALSE；如果线性表L不存在，返回INFEASIBLE。

**⑸求表长：函数名称是ListLength(L)；**

如果线性表L存在，返回线性表L的长度，否则返回INFEASIBLE。

**⑹获得元素：函数名称是GetElem(L,i,e)；**

如果线性表L存在，获取线性表L的第i个元素，保存在e中，返回OK；如果i不合法，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

**⑺查找元素：函数名称是LocateElem(L,e,compare())；**

如果线性表L存在，查找元素e在线性表L中的位置序号并返回OK；如果e不存在，返回ERROR；当线性表L不存在时，返回INFEASIBLE。

**⑻获得前驱：函数名称是PriorElem(L,cur\_e,pre\_e)；**

如果线性表L存在，获取线性表L中元素e的前驱，保存在pre中，返回OK；如果没有前驱，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

**⑼获得后继：函数名称是NextElem(L,cur\_e,next\_e)；**

如果线性表L存在，获取线性表L元素e的后继，保存在next中，返回OK；如果没有后继，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

**⑽插入元素：函数名称是ListInsert(L,i,e)**；

如果线性表L存在，将元素e插入到线性表L的第i个元素之前，返回OK；当插入位置不正确时，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

**⑾删除元素：函数名称是ListDelete(L,i,e)；**

如果线性表L存在，删除线性表L的第i个元素，并保存在e中，返回OK；当删除位置不正确时，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

**⑿遍历表：函数名称是ListTraverse(L,visit())；**

如果线性表L存在，依次显示线性表中的元素，每个元素间空一格，返回OK；如果线性表L不存在，返回INFEASIBLE。

1. **将数据元素写入文件：函数名称是SaveList(L,FileName[]）；**

如果线性表L存在，将线性表L的的元素写到FileName文件中，返回OK，否则返回INFEASIBLE。

1. **将文件数据写入线性表：函数名称是LoadList（L，FileName[]）；**

如果线性表L存在，就将FileName文件中的数据写入线性表中，返回OK。否则返回INFEASIBLE；

1. **添加线性表：函数名称是AddList（Lists，ListName[]）;**

只需要在Lists中增加一个名称为ListName的线性表,并返回OK，如果添加的线性表个数超过数组范围，返回ERROR。

1. **删除线性表：函数名称是RemoveList（Lists，ListName[]）；**

在Lists中删除一个名称为ListName的线性表；如果不存在该线性表，返回ERROR；删除成功返回OK。

1. **查找线性表：函数名称是LocateList（Lists，ListName[]）；**

在Lists中查找一个名称为ListName的线性表，成功返回逻辑序号，否则返回0；

**下面是附加功能：**

1. **对线性表进行升序排序：函数名称是UpList（L）；**

如果线性表存在，就对线性表进行升序排序并返回OK；否则返回INFEASIBLE.

1. **对线性表进行降序排列：函数名称是DownList（L）；**

如果线性表存在，就对线性表进行降序排列并返回OK，否则返回INFEASIBLE.

1. **判断子表：函数名称为SubList（L,S);**

如果L和S两个线性表都存在，判断S是不是L的子集；否则返回INFEASIBLE。

1. **求交集：函数名称是InterList（L，S,p）**

如果L，S，p三个线性表都存在，将L与S 两个顺序线性表的共同元素保存在p线性表中；否则返回INFEASIBLE。

1. **求两线性表的并集：函数名称是SumList（L,S,p）；**

如果L，S，p三个线性表都存在，将L与S 两个顺序线性表的元素保存在p线性表中，保存后p也为顺序线性表；否则返回INFEASIBLE。

1. **求差集；函数名称为DifferList（L,S,p）；**

如果L，S，p三个线性表都存在，将L与S 两个顺序线性表的不存在于另一个线性表中的元素保存在p线性表中，保存后p也为顺序线性表；否则返回INFEASIBLE。

**1.3系统实现**

函数实现：

**（1）初始化表：函数名称是InitList(L)；**

如果存在线性表，如果再构造线性表，会覆盖掉原先的元素，所以返回INFEASIBLE；

如果该线性表之前不存在，调用malloc函数给线性表分配空间，如果分配失败，就返回INFEASIBLE；如果分配成功，就对线性表进行初始化，length初始化为0，listsize初始化为前面已经定义的最大值。

**（2）销毁表：函数名称是DestroyList(L)；**

如果线性表不存在，无法进行销毁操作，返回INFEASIBLE；

如果线性表存在，首先清空线性表，即L.length=0;然后free函数释放线性表所分配的内存。重点：free掉以后一定要将指针指向NULL，不然野指针会出问题！

**（3）清空表：函数名称是ClearList(L)；**

如果线性表不存在，无法进行清空操作，就返回INFEASIBLE；

如果线性表存在，就将线性表清空，即将线性表的长度的长度变为0，即L.length=0;并返回OK以表示线性表被清空成功。

**（4）判定空表：函数名称是ListEmpty(L)；**

如果线性表不存在，无法进行判空操作，就返回INFEASIBLE；

如果线性表存在，判断线性表是否为空，如果L.length=0，那么线性表为空，返回TRUE，如果L.length不等于0，那么线性表不为空，返回FALSE。

**（5）求表长：函数名称是ListLength(L)；**

如果线性表不存在，无法进行求表长的操作，就返回INFEASIBLE；

如果线性表存在，就将L.length作为返回值将表长返回。

**（6）获得元素：函数名称是GetElem(L,i,e)；**

如果线性表不存在，无法进行获取元素的操作，返回INFEASIBLE；

如果线性表已存在，判断i值是否合法，如果i<=0或者i>L.length,线性表中只有L.length个元素，此时无法获取到第i个元素，返回ERROR； 如果i>0并且i<=L.length，那么就将elem数组中的第i个元素赋值给e并且返回OK以表示函数结束。

**（7）查找元素：函数名称是LocateElem(L,e,compare())；**

如果线性表不存在，无法进行查找元素的操作，返回INFEASIBLE；

如果线性表已存在，对线性表进行遍历，从第一个元素开始查找，如果有与e相等的元素，就直接返回该元素所处的位置，如果遍历结束还没有e元素，就返回0.

**（8）获得前驱：函数名称是PriorElem(L,cur\_e,pre\_e)；**

如果线性表不存在，无法进行查找元素的操作，返回INFEASIBLE；

如果线性表已存在，对线性表中具有前驱的元素进行遍历，从第一个具有前驱的元素开始查找，即第二个元素到最后一个元素，如果有与e相等的元素，就直接返回该元素的前驱所处的位置，如果遍历结束还没有e元素，就返回0.

1. **获得后继：函数名称是NextElem（L,E,Next）；**

如果线性表不存在，无法进行查找元素的操作，返回INFEASIBLE；

如果线性表已存在，对线性表中具有后继的元素进行遍历，从第一个具有后继的元素开始查找，即第一个元素到倒数第二个元素，如果有与e相等的元素，就直接返回该元素的后继所处的位置，如果遍历结束还没有e元素，就返回0.

1. **插入元素：函数名称是ListInsert(L,i,e)；**

如果线性表不存在，无法进行获取元素的操作，返回INFEASIBLE；

如果线性表已存在，判断i值是否合法，如果i<=0或者i>L.length+1,线性表中只有L.length个元素，插入的元素只能在第1到L.length+1的元素前面插入，其中L.length+1为在最后一个位置插入元素，此时i值不合法，返回ERROR；

如果i值合法，判断线性表是否已满，如果已满，调用realloc函数对线性表的内存重新分配，如果分配失败，返回OVERFLOW；

如果成功或者本身线性表未满，从第i个元素开始，之后的元素每个都后移一位然后将e元素插入第i位，最后使线性表长度加一更新值。

1. **删除元素：函数名称是ListDelete(L,i,e)；**

如果线性表不存在，无法进行删除元素的操作，返回INFEASIBLE；

如果线性表已存在，判断i值是否合法，如果i<=0或者i>L.length,线性表中只有L.length个元素，此时无法找到并删除第i个元素，返回ERROR；

如果i>0并且i<=L.length，那么就将elem数组中的第i个元素删除，删除操作为，从第i+1位值开始，到最后一位，都往前移一位以删除该值，并且使线性表长度减一，最后返回OK以表示函数结束。

1. **遍历表：函数名称是ListTraverse(L,visit())；**

如果线性表不存在，无法进行删除元素的操作，返回INFEASIBLE；

如果线性表已存在，遍历线性表，从第一个元素开始到最后一个，逐个输出，注意空格的输出最后一个元素后面不需要输出空格。

1. **将数据元素写入文件：函数名称是SaveList(L,FileName[]）；**

如果线性表不存在，无法进行元素读入文件的操作，返回INFEASIBLE； 如果线性表已存在，首先以“w”的方式打开名为filename的文件，如果打开失败，直接返回ERROR；

打开成功后，开始遍历线性表，从第一个元素开始，逐个向文件中写入数据，调用fprintf函数；遍历完成后关闭文件。

注意，每次写入文件时，数据与数据之间应该有间隔，不能将数据直接全部输入，不然读取文件的时候，无法读取到单个的数据。

1. **将文件数据写入线性表：函数名称是LoadList（L，FileName[]）；**

如果线性表不存在，无法进行文件元素写入线性表的操作，返回INFEASIBLE；

如果线性表已存在，首先以“r”的方式打开名为filename的文件，如果打开失败，直接返回ERROR；

打开成功后，开始遍历文件，从第一个元素开始，逐个向文件中写入数据，调用fscanf函数；然后调用ListInsert函数将读出的元素插入到线性表的末尾中，遍历完成后关闭文件。

1. **添加线性表：函数名称是AddList（Lists，ListName[]）;**

如果多线性表集里面的线性表个数已达到10，已无法再插入线性表，因为这个数组使静态的；

如果数组未满，就向多线性表中插入一个线性表，首先将ListName复制到新增的线性表的名称中去，while循环遍历到名称结尾进行赋值来复制。

调用InitList函数对最后一个线性表后面的数组元素中的线性表进行初始化，然后调用ListInsert函数向该线性表中插入元素，线性表加入成功后，多线性表的长度加一。

1. **删除线性表：函数名称是RemoveList（Lists，ListName[]）；**

首先遍历多线性表，查看是否有名称为ListName的线性表，采用while循环遍历查看线性表名称与指定名称是否相等，如果有不相等的元素，直接break表示不相等，如果两字符串都到达了结尾，说明名称相等，

然后进行删除线性表操作；即从该元素下一个开始，到最后一个，每个元素都前移一位，然后销毁最后一个线性表，并且使多线性表的长度减一。

1. **查找线性表：函数名称是LocateList（Lists，ListName[]）；**

遍历多线性表，查看是否有名称为ListName的线性表，采用while循环遍历查看线性表名称与指定名称是否相等，如果有不相等的元素，直接break表示不相等，如果两字符串都到达了结尾，说明名称相等，此时直接可返回该线性表的位置，如果遍历完整个多线性表没有相等的名称，就返回0以表示无。

**附加功能：**

1. **对线性表进行升序排序：函数名称是UpList（L）；**

如果线性表不存在，无法进行对元素排序的操作，返回INFEASIBLE；如果线性表已存在，采用冒泡排序法排序。

1. **对线性表进行降序排列：函数名称是DownList（L）；**

如果线性表不存在，无法进行对元素排序的操作，返回INFEASIBLE；如果线性表已存在，采用冒泡排序法排序。

1. **判断子表：函数名称为SubList（L,S);**

如果俩线性表有一个不存在，就无法进行比较操作，返回INFEASIBLE；

如果两线性表都存在，判断S是否为L的一个子集，遍历S中的元素，看是否在L这个线性表中存在，即可调用LocateList函数判断该元素是否在L这个线性表中，如果遍历过程中，有元素不在L中，直接返回ERROR以表示不是子集，如果遍历结束,S的元素在L 中都出现了，那么S是L的一个子集。

1. **求交集：函数名称是InterList（L，S,p）；**

如果俩线性表有一个不存在，就无法进行比较操作，返回INFEASIBLE；

如果两线性表都存在，遍历两个线性表，由于是顺序表，则可以通过比较得到相等的元素，

首先对L中的元素进行过滤，将比当前的S所拿出的元素小的元素都过滤掉，即i++，之后再过滤S中的元素，遇到相等的元素，就将该元素插入到p线性表中，调用ListInsert函数，直到遍历到其中一个线性表到达结尾结束。

1. **求两线性表的并集：函数名称是SumList（L,S,p）；**

如果俩线性表有一个不存在，就无法进行比较操作，返回INFEASIBLE；

如果两线性表都存在，开始求并集。俩顺序表的并集也是顺序表，

首先令i，j表示当前遍历到的L,S的线性表的位置，首先插入比较小的元素，先遍历L，插入比当前的S中的第j位小的元素，即i++；之后再遍历S，插入比当前的L中的第i位小的元素，即j++；如果遇到两线性表中相等的元素，插入该元素并且需要i++，j++；直到两个线性表都结束。

**（23）求差集；函数名称为DifferList（L,S,p）；**

如果俩线性表有一个不存在，就无法进行比较操作，返回INFEASIBLE；

如果两线性表都存在，开始求差集。俩顺序表的差集也是顺序表，

首先令i，j表示当前遍历到的L,S的线性表的位置，首先插入比较小的元素，先遍历L，插入比当前的S中的第j位小的元素，即i++；之后再遍历S，插入比当前的L中的第i位小的元素，即j++；如果遇到两线性表中相等的元素，不插入并且需要i++，j++；直到两个线性表都结束。

**1.4系统测试**

**1.4.1 系统测试目的**

通过运行每个程序功能来检测各个功能实现正确性，一方面检测正常数据是否能够正常处理，另一方面要检测异常数据处理并检验程序是否能够判断该数据不合法且给予提示，此过程程序能继续运行不受影响。

**1.4.2 系统测试结果**

表1-1

|  |  |  |  |
| --- | --- | --- | --- |
| 选择操作 | 输入 | 预计输出 | 实际输出 |
| 创建线性表 | 1 | 线性表创建成功 |  |
| 销毁线性表 | 2 | 线性表销毁成功 |  |
| 销毁线性表 | 2 | 线性表本身不存在，销毁失败 | Q~RYGF6}NH9(IQJX3P_U%85 |
| 创建线性表 | 1 | 线性表创建成功 |  |
| 清空线性表 | 3 | 线性表已清空 | Z{%(1FT[J3K~KE5XIH(FX(1 |
| 读入文件 | 14 | 文件数据读入线性表成功 | ~U29YGZRTC]LJWCR9RK6Y25 |
| 获取表长 | 5 | 线性表长度为13 | 6WPDFVIT]_CY]UV_ZI[[4CT |
| 获取元素 | 6 1 | 线性表第1个值是1 | HQ$)2AAL%ZB2AIKR~_17M9N |
| 遍历线性表 | 12 | 1 2 3 4 5 6 7 8 9 10 23 24 0 | )P2CW__66JZEUUR]4{30X}5 |
| 获取位置 | 7 10 | 元素10在线性表中的位置位10 | 06I@VBP(4T2%V8Q6V%DX[}B |
| 获得后继 | 9  9 | 线性表中元素9后继元素为10 | E}OSU0D[}$T9GRNZ%GA$RH3 |
| 获得前驱 | 8  9 | 线性表中元素9的前驱元素为8 | _242TZ@K85W6ZG0GRL)}CJF |
| 判空 | 4 | 线性表非空 | EIAK(Q8`A@@(K__G57M4TUL |
| 获取表长 | 5 | 线性表的长度为13 | V0U{47PYFEP(TT~Q0@`%5@X |
| 插入元素 | 10  5 11 | 插入成功 | T24WLD9F~S7RL~I_R7][Q)9 |
| 遍历 | 12 | 1 2 3 4 11 5 6 7 8 9 10 23 24 0 | Y]{HQP_XS[V}WOO2[L691DT |
| 删除元素 | 13  7 | 删除成功，删除元素为6！ | }A_QNW7802]{`BAPX7M`C0O |
| 遍历线性表 | 12 | 1 2 3 4 11 5 7 8 9 10 23 24 0 | MABKYEDQ4}9PGAI(PM1(FDI |
| 写入文件 | 13  data.txt | 写入文件成功 | P)9I2}L@H7ZKU1NB}DVHEWN |
| 添加线性表 | 15 | 给出多个线性表的遍历：  yi 1 6 9  Er 6 8 11 45  San 3 7 9 23 67 | G_TCP9F79QBW$2PQ%FOQE}B |
| 查找线性表 | 17  er | er线性表的位置为2 | }0TEXSBT4[7P[MGRZ${ZB0P |
| 查找线性表 | 17  www | 不存在该线性表 | O@`HU_YY4]NZ@JLRPYB4%NN |
| 线性表升序 | 18 | 升序后的线性表为： | _]HNN${[MUBWGH}~RHGE$TO |
| 降序排列 | 19 | 降序后的线性表：24 23 11 10 9 8 7 5 4 3 2 1 | L%0BP002MP`UX80])HNYPQ3 |
| 删除线性表 | 16  er | er线性表删除成功 | KB836(9}APRWAJVL]RCX}MC |
| 退出 | 0 | 欢迎下次再使用本系统 | %@]L{9K@5NPFEIB@2U8L1V1 |
| 添加线性表 | 15 | 给出多个线性表的遍历：  yi 5 8 9  Er 8 9 12 24  San 8 23 | 9J0WFIA}@GV7Y~Q]QEOL]US |
| 判断子集 | 20 | 二者不存在子集关系 | O_@NC{]Z`_@4W291_$KNI{A |
| 多线性表取交集 | 21 | 两线性表的交集为8 9 | L5Y6@FG52H8X~)B8%L}MX9K |
| 多线性表取差集 | 23 | 两线性表的差集为5 12 24 | W@Q%I`)RQ42G_{D]$54_3@C |
| 多线性表取并集 | 22 | 两线性表的并集是5 8 9 12 24 | R21MEG{L%YSH1SIQ5AT4O]N |
| 退出 | 0 | 欢迎下次再使用本系统 | %@]L{9K@5NPFEIB@2U8L1V1 |

**1.5实验小结**

本次实验是为了让我们对线性表有一个更深层次的理解，其中包含了一些对线性表的基本功能的实现，比如初始化线性表，删除插入元素等对单个线性表的操作以及对多线性表的操作。基础功能是我们必须要掌握的功能，我们应当熟练掌握，在实验过程中，我发现，在一些小细节上面需要多加注意，比如指针在初始时是否指为空，在释放过后是否指为空，如果不指为空，程序可能会出现不可避免的错误，在进行删除或者插入操作时，对指针的操作需要多加注意。另外一些length上的增减操作需要多加注意。

在附加功能中，我基于字符串的一些基础操作，想到了线性表的子集、交集、并集、差集等操作。这些功能不算复杂，也运用了一些基础功能的函数。所以，基础功能的函数是基础，我们需要熟练掌握，才能组合出更复杂的功能。

这次的线性表实验整体来说还是比较简单的，系统的功能并不复杂，代码也不是很复杂，但是也让我们第一次认识到了系统是如何实现的，以一种简单的方式开始实验，为以后打基础，以后还需要更多的努力。

**2 基于链式存储结构的线性表实现**

**2.1问题描述**

通过实验达到

⑴加深对线性表的概念、基本运算的理解；

⑵熟练掌握线性表的逻辑结构与物理结构的关系；

⑶物理结构采用单链表,熟练掌握线性表的基本运算的实现。

**2.2系统设计**

**2.2.1整体结构设计**

实现顺序储存的线性表的基本运算操作。通过switch语句根据菜单系统上的提示内容输入数字进行对应线性表操作，依据最小完备性和常用性相结合的原则，以函数形式定义了线性表的初始化表、销毁表、清空表、判定空表、求表长和获得元素等12种基本运算，包括：创建，销毁，清空，判空，求表长，获取元素，查找元素，获得前驱，获得后继，插入元素，删除元素，遍历线性表，数据读写文件，及相关附加功能：多线性表管理（增加一个新线性表，移除一个线性表，查找线性表）、排序、逆置，归并。

**2.2.2数据结构设计**

线性表基本运算设计：

**(1)初始化表：函数名称是InitList(L)；**

初始条件是线性表L不存在；操作结果是构造一个空的线性表。

**(2)销毁表：函数名称是DestroyList(L)**；

初始条件是线性表L已存在；操作结果是销毁线性表L。

**(3)清空表：函数名称是ClearList(L)；**

初始条件是线性表L已存在；操作结果是将L重置为空表。

**(4)判定空表：函数名称是ListEmpty(L)；**

初始条件是线性表L已存在；操作结果是若L为空表则返回TRUE,否则返回FALSE。

**(5)求表长：函数名称是ListLength(L)；**

初始条件是线性表已存在；操作结果是返回L中数据元素的个数。

**(6)获得元素：函数名称是GetElem(L,i,e)；**

初始条件是线性表已存在，1≤i≤ListLength(L)；操作结果是用e返回L中第i个数据元素的值。

**(7)查找元素：函数名称是LocateElem(L,e,compare())；**

初始条件是线性表已存在；操作结果是返回L中第1个与e满足关系compare（）关系的数据元素的位序，若这样的数据元素不存在，则返回值为0。

**(8)获得前驱：函数名称是PriorElem(L,cur\_e,pre\_e)；**

初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是第一个，则用pre\_e返回它的前驱，否则操作失败，pre\_e无定义。

**(9)获得后继：函数名称是NextElem(L,cur\_e,next\_e)；**

初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是最后一个，则用next\_e返回它的后继，否则操作失败，next\_e无定义。

**(10)插入元素：函数名称是ListInsert(L,i,e)**；

初始条件是线性表L已存在，1≤i≤ListLength(L)+1；操作结果是在L的第i个位置之前插入新的数据元素e。

**(11)删除元素：函数名称是ListDelete(L,i,e)；**

初始条件是线性表L已存在且非空，1≤i≤ListLength(L)；操作结果：删除L的第i个数据元素，用e返回其值。

**(12)遍历表：函数名称是ListTraverse(L,visit())；**

初始条件是线性表L已存在；操作结果是依次对L的每个数据元素调用函数visit()。

**（13）将数据元素写入文件：函数名称是SaveList(L,FileName[]）；**

初始条件是线性表L存在并非空；操作结果：将线性表L的的元素写到FileName文件中。。

**（14）将文件数据写入线性表：函数名称是LoadList（L，FileName[]）；**

初始条件是线性表L；操作结果：将FileName文件中的数据写入线性表中。

**（15）添加线性表：函数名称是AddList（Lists，ListName[]）;**

初始条件是多线性表数组未满；操作结果：在多线性表中添加一个名称为listname的线性表。

**（16)删除线性表：函数名称是RemoveList（Lists，ListName[]）；**

初始条件是多线性表非空并且存在名称为listname的线性表；操作结果：删除多线性表中名称为listname的线性表。

**(17)查找线性表：函数名称是LocateList（Lists，ListName[]）；**

初始条件是多线性表非空并且存在名称为listname的线性表；操作结果：返回该线性表的位置序号。

**下面是附加功能：**

**(18)对线性表进行升序排序：函数名称是UpList（L）；**

初始条件为线性表存在；操作结果：线性表的顺序为升序。

**(19)对线性表进行降序排列：函数名称是DownList（L）；**

初始条件是线性表存在并非空；操作结果：线性表的顺序为降序。

1. **对线性表进行逆置：函数名称为ReverseList(L)；**

初始条件是线性表存在并非空；操作结果：线性表逆序排列。

1. **对两个线性表进行归并：函数名称为MergeList(L,b)；**

初始条件是多线性表中含有至少两个线性表并且为升序；操作结果：两个顺序线性表归并到第一个线性表中得到一个顺序线性表并且去除相同元素。

**2.3系统实现**

函数实现：

1. **InitList(LinkList &L):**

如果线性表存在，返回INFEASIBLE；

如果线性表不存在，对头结点分配空间；使L的next指针指空；返回OK。

**（2）DestroyList(LinkList &L)：**

如果线性表不存在，返回INFEASIBLE；

令a为当前要释放空间的结点，b为后继结点；

While循环释放当前结点a的空间，a，b结点不断后移，直到a为空；

最后对L赋值为空，防止野指针产生。

**（3）ClearList(LinkList &L)：**

如果线性表不存在，返回INFEASIBLE；

初始化a为首元结点，如果a为空，直接返回OK已清空；如果a不为空，令b为a的后继结点。

While循环，释放当前结点的空间，之后a，b两个结点都后移，直到a为空结束循环；

最后使头结点的next为空表示首元结点为空。

**（4）ListEmpty(LinkList L)**

如果线性表不存在，返回INFEASIBLE；

如果首元结点的next指针不为空，返回FASLE表示线性表非空；反之，返回OK，线性表为空。

**（5）ListLength(LinkList L)**

如果线性表不存在，返回INFEASIBLE；

初始化a为首元结点，len初始化为0。

While循环，判断条件为a是否为空，a空时结束循环；每次进入循环，len增一表示当前的结点含有元素，线性表长度加一；a指向下一节点再次进入循环。

返回len表示线性表的长度。

**（6）GetElem(LinkList L,int i,ElemType &e)**

如果线性表不存在，返回INFEASIBLE；

判断i值的合法性，若i<=0，位置序号的赋值不合法，返回ERROR；

初始化a为首元结点，len为当前元素所在位置；

While循环，判断条件为a是否为空，a为空时停止计数，线性表遍历结束；每次循环：len增一以表示当前结点所处的线性表位置；如果len＝i，用e储存该位置的元素并返回OK；a指向下一个节点后继续下一轮循环。

如果循环结束并未找到指定位置的值，表示i值输入的不合法，返回ERROR。

**（7）LocateElem(LinkList L,ElemType e)**

如果线性表不存在，返回INFEASIBLE；

初始化a为首元结点，i为0；while循环判断条件为a是否为空，即遍历结束；若a不为空，i++表示当前结点的位置，判断a->data是否等于e，如果相等，返回位置序号i，如果不相等，a指向下一个结点，进入下一个循环；

如果遍历结束没有相等的e，就返回ERROR表示不含有e元素。

**（8）PriorElem(LinkList L,ElemType e,ElemType &pre)**

如果线性表不存在，返回INFEASIBLE；

如果线性表为空或者只有一个首元结点，表示没有具有前驱元素的元素；返回ERROR；

初始化a为首元结点，b为第二个结点；即b是第一个有前驱的结点；

While循环，循环条件为b不为空；每次循环：判断b的元素是否为条件判断值，如果是，将b的前驱a的元素赋给e并返回ok；如果不是，a，b均指向各自的下一结点重新循环；

如果遍历结束无这样的b，返回ERROR表示该元素在线性表中无前驱。

**（9）NextElem(LinkList L,ElemType e,ElemType &next)**

如果线性表不存在，返回INFEASIBLE；

如果线性表为空或者只有一个首元结点，表示没有具有后继元素的元素；返回ERROR；

初始化a为首元结点；

While循环，循环条件为a的后继不为空；每次循环：判断a的元素是否为条件判断值，如果是，将a的后继a->next的元素赋给e并返回ok；如果不是，a指向各自的下一结点重新循环；

如果遍历结束无这样的a，返回ERROR表示该元素在线性表中无前驱。

**（10）ListInsert(LinkList &L,int i,ElemType e)**

如果线性表不存在，返回INFEASIBLE；

如果输入的插入位置小于1，必定不合法，返回ERROR；合法值为1-length+1；

初始化a为头结点，len为0；while循环，判断条件为a是否为空，若a不为空，进入循环；判断len是否等于i-1，插入到第i个元素之前也就是插入到第i-1个元素之后，若相等，即找到了第i-1个元素，此时为插入元素分配一个结点空间b，b->data=e，让b的next等于a的next，a的next指向b，完成插入，返回OK；若不想等，a指向下一个结点，len++；表示a的位置；

若遍历结束并为找到该位置，说明输入的i值不合法，返回ERROR；

1. **ListDelete(LinkList &L,int i,ElemType &e)**

如果线性表不存在，返回INFEASIBLE；

如果输入的删除位置小于1，必定不合法，返回ERROR；合法值为1-length；

初始化a为头结点，len为0；

while循环，判断条件为a->next是否为空，若a->next不为空，说明删除的位置合法，进入循环；判断len是否等于i-1，删除第i个元素，也就是让第i-1个元素的next记录第i+1个元素；如果相等，也就是删除该元素后面的元素，用e记录a->next->data，a->next指向下一个结点，然后释放a原本后面的结点的空间，并返回OK；如果不相等，a指向下一结点，len++表示a的位置；

如果遍历结束并没有这样的值，返回ERROR。

1. **ListTraverse(LinkList L)**

如果线性表不存在，返回INFEASIBLE；

初始化a为首元结点；

While循环，判断条件为a是否为空；输出a->data;判断a之后是否还有元素，如果有，输出空格，如果没有，不输出空格；a指向下一个结点，再次循环；

遍历完毕，返回ERROR。

1. **SaveList(LinkList L,char FileName[])**

如果线性表不存在，返回INFEASIBLE；

初始化a为首元结点，file为以“w”的方式打开，如果打开失败，返回ERROR；

While循环，判断a是否为空，fprintf函数将数据+一个空格写入文件；然后a指向下一个结点后继续循环；

遍历完毕后关闭文件指针并返回OK。

1. **LoadList(LinkList &L,char FileName[])**

如果L存在，调用ClearList函数清空L；如果L不存在，调用InitList函数初始化L；

初始化a为头结点，b为NULL；以“r”的方式打开文件，如果打开失败，返回ERROR；

While循环，判断条件是读取数据是否到达文件末尾；为b分配空间，数据域赋值从文件中读取的数据，指针域为空，a指针域更新为b，然后a，b指针均后移；直到达到文件末尾；

遍历结束，关闭文件，返回OK。

下面是附加功能：

1. **AddList(LISTS &lists,char ListName[])**

如果数组已满，无法再加入线性表，返回ERROR；

将要加入的线性表名字复制到多线性表最后一个线性表后面的线性表中去，调用InitList函数初始化线性表，然后向其中插入元素。

插入成功后，lists的长度增一，返回OK。

1. **RemoveList(LISTS &lists,char ListName[])**

初始化i为多线性表的长度，j为0；

For循环j从0到i-1，比较线性表的名称跟指定名称，如果相等，删除该位置的线性表，也就是从该位置的后一位开始每位都前移一个；

删除完毕，长度减一，返回OK；

如果遍历结束时未找到相同名称的，返回ERROR。

1. **LocateList(LISTS lists,char ListName[])**

初始化i为0；

For循环，i从0到length-1；比较线性表名称与指定名称，如果相等，直接返回位置i+1；

遍历结束如果没有与指定名称相等的线性表，返回ERROR。

1. **UpList(LinkList &L)**

如果L不存在，返回INFEASIBLE;

For循环，p从首元结点开始，到最后一个结点；q从当前的p结点开始，一直到最后一个结点，与p的数据域比较，找到最小值，然后交换最小值的数据域与当前节点的数据域；

遍历结束，返回ERROR。

1. **DownList(LinkList &L)**

如果L不存在，返回INFEASIBLE;

For循环，p从首元结点开始，到最后一个结点；q从当前的p结点开始，一直到最后一个结点，与p的数据域比较，找到最大值，然后交换最小值的数据域与当前节点的数据域；

遍历结束，返回ERROR。

1. **ReverseList(LinkList &L)**

如果L不存在，返回INFEASIBLE;

初始化now记录当前到达的结点的位置，pre记录逆置以后now该指向的结点 ；

While循环，判断条件为now是否为空，即是否遍历完毕；next记录当前结点now的下一个结点，对now的指针域进行更新，指向其前驱；完毕后pre，now均后移继续循环；

循环结束后，将头结点的指针域更新，指向pre；返回OK。

1. **MergeList(LinkList &L,LinkList b)**

如果要归并的两个线性表不存在，返回INFEASIBLE;

初始化a为第一个线性表的首元结点，b为第二个线性表的首元结点；

While循环，判断条件是a，b都不为空；进入循环后，比较a，b的数据域，如果a的数据域更小，就使a以及a的前驱pre后移；如果相等，b后移结点；如果b小a大，就在pre和a之前插入b结点；插入以后，b后移至第二个线性表的下一个结点，pre后移，再次循环；

循环完毕后，如果a为NULL，可能第二个线性表并未插入完毕，此时后面的数据都是较大的，所以直接让最后一个结点的指针域指向剩余结点即可，最后一个结点为pre，剩余结点以b开头；

归并完毕，返回OK。

**2.4系统测试**

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 实验输入 | 预期结果 | 实际结果 |
| 1 | 1 | 线性表创建成功 | $O7X7JHV[8K[XF`X~U6T(D6 |
| 2 | 2 | 线性表销毁成功 | )G}`I1~BXCW@4{)39D)NFLW |
| 3 | 3 | 线性表不存在 | ZB3Z8G55C4FQO3WA5WTQPMT |
| 4 | 1 | 线性表创建成功 | V(4KET9[SQL`AR6{PLP(LUD |
| 5 | 5 | 线性表长度为0 | Y3LQVU@]]}3@)E`QRA9_S`9 |
| 6 | 4 | 线性表为空 | RRKTJS7RU%K[UU]V9(WK%TJ |
| 7 | 6  1 | 输入的1值不合法 | 9G)5GESJ7@M52D(`UO)$%@I |
| 8 | 10  1 2 | 插入成功 | )5~MAMZ(20`_P69{~Q110NK |
| 9 | 10  2 23 | 插入成功 | 5PZ[Z8QYYI(3FJ[Z5]}F{L7 |
| 10 | 10  3 11 | 插入成功 | ZAZYOWJV`LTWX]IQ3O]8~T8 |
| 11 | 10  5 10 | 输入的5值不合法 | VA`BJSI6X6AULPDH_B2]6%0 |
| 12 | 10  4 10 | 插入成功 | XL3PN%5L$RN2K[$2LM5Z[W2 |
| 13 | 10  5 20 | 插入成功 | TI)}BLJYBKB$TVDRTHZ[}X1 |
| 14 | 10  5 20 | 插入成功 | 7}(QS~S11}{R{9IR6QT7@26 |
| 15 | 12 | 2 23 11 10 20 10 | DQ)IHF5H~Z`}965}[]MQE$Y |
| 16 | 11  6 | 删除成功，删除元素为10！ | JS}`PHTAI`ZIQTS$)O2RX0P |
| 17 | 12 | 2 23 11 10 20 | V0G$`OO~Z8]YX%Y7PHI7RTG |
| 18 | 10  6 99 | 插入成功 | VTYSZ37{(ZST_XBBO2EO@FV |
| 19 | 10  7 223 | 插入成功 | N_2VRZ9Q_R5TBH~KVO%GRJS |
| 20 | 12 | 2 23 11 10 20 99 223 | 0D%M{6PBXE%MX4P%A$YCI90 |
| 21 | 10  7 6 | 插入成功 | BXK58)$2]L@$UJE5{{@KCAJ |
| 22 | 12 | 2 23 11 10 99 6 223 | ]IQ1E8BBL9~9FYMWA6A8U2Y |
| 23 | 13  xxx.txt | 写入文件成功 | W~[)[`])(4O(PX{6BGA43E4 |
| 24 | 3 | 线性表已清空 | {IV{8[W)RYKSL_TX}}M)_1R |
| 25 | 14  xxx.txt | 文件数据写入线性表成功 | )%%ENOKZ)}F(86)4IMN%XA2 |
| 26 | 12 | 2 23 11 10 20 99 6 223 | }`LUB35W72E4@_B%S9Z}~YX |
| 27 | 5 | 线性表长度为8 | K%3Z0U}T6ECDRHG42}78JJI |
| 28 | 6  6 | 线性表第6个值为99 | @XYF172P`5_ZT5%@[6HJ$SU |
| 29 | 7  23 | 元素23在线性表中的位置为2 | NLG_H%L2{_RQVFH{BA%05WX |
| 30 | 7  88 | 线性表中不存在该元素 | ~~](RG~}~]67}NAW@9K9Y8S |
| 31 | 8  11 | 线性表中元素11的前驱元素为23 | JRGH%E5EW2C]RT3LSVAFP$9 |
| 32 | 8  89 | 没有前驱元素 | R]@8OVB}EA{`K)WRT0J(@IK |
| 33 | 9  6 | 线性表中元素6的后继元素为223 | @L(MI30NK2KU{~SLQAD[$QK |
| 34 | 9  223 | 没有后继元素 | T5[W)PU)0VD{NFWJW{{9XXS |
| 35 | 18 | 升序后的线性表：2 6 10 11 20 23 99 223 | 7WDN2VJ8EPE9P%826@N@S44 |
| 36 | 19 | 降序后的线性表：223 99 23 20 11 10 6 2 | W%EVAGVTKR()ASSTYUQL(}M |
| 37 | 20 | 逆置后的线性表：  2 6 10 11 20 23 99 223 | D{N8{P3%K)~YR82JY[7RTSE |
| 38 | 15 | 线性表的遍历 | GMWXB]~B_Q8L96((J{B5`ZN |
| 39 | 17  Er | Er线性表的位置为2 | 2)NISRY5[FC3I[J88A_9QVR |
| 40 | 17  San | San线性表的位置为3 | 0XN$4WY)Z2CJ}G1G6LHT3BC |
| 41 | 17  Si | 不存在该线性表 | {$HI54C89ZM1MHLLQ_8GLCV |
| 42 | 16  San | San线性表删除成功 | MN15MPB6UR`%S}KPEMJA8JS |
| 43 | 16  San | 删除失败 | )XISR9UGT}61AFTNQC3W)Z7 |
| 44 | 21 | 归并后的线性表为：1 2 3 4 5 6 7 9 11 | R8G[HONQXBTY2J6I]B7D}J8 |
| 45 | 0 | 欢迎下次使用本系统 | LYO_SC)~0)YEKUSYZVQ8P@K |

**2.5实验小结**

这次的实验和第一次实验难度相当，都不算很难，上次是基于顺序储存结构的线性表系统实现，这次是基于链式结构的线性表系统实现。

与上一个的实验的区别基本在与物理结构不同，上一个实验是对动态数组进行操作，这次是对动态链表进行操作，后者在插入、删除等方面的操作上面更简单，所需要的时间复杂度和空间复杂度都比较小，但是前者在查找指定位置元素以及初始化和销毁方面比较方便，两种结构都有各自的应用场景以及优缺点。

在链式储存中，需要注意的是前驱的记录和后继next的使用；此外还需要注意一些细节，最后循环结束时，尾结点是哪个指针；还需要注意的是NULL的使用，next野指针应该赋空。

而附加功能是在基于之前c语言对链表的学习结合了数据结构做的，这次的排序由于数据域较短，所以交换数据域，在数据域长的时候，交换结点是更方便的，以及归并，两个顺序表的归并还是顺序表，所以，要注意结点后移开始和结束的边界条件，这点容易出问题。

附录A 基于顺序存储结构线性表实现的源程序

/\* Linear Table On Sequence Structure \*/

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

/\*---------page 10 on textbook ---------\*/

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define OVERFLOW -2

typedef int status;

typedef int ElemType; //数据元素类型定义

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

typedef struct{ //顺序表（顺序结构）的定义

ElemType \* elem;

int length;

int listsize;

}SqList;

typedef struct{ //线性表的集合类型定义

struct { char name[30];

SqList L;

} elem[10];

int length;

}LISTS;

LISTS Lists;

/\*-----page 19 on textbook ---------\*/

status InitList(SqList &L);//初始化一个线性表

status DestroyList(SqList &L);//销毁一个线性表

status ClearList(SqList &L);//清空一个线性表

status ListEmpty(SqList L);//判断一个线性表是否为空

int ListLength(SqList L);//看一个线性表的长度

status GetElem(SqList L,int i,ElemType &e);//取出第i个元素

status LocateElem(SqList L,ElemType e); //定位e元素的位置

status PriorElem(SqList L,ElemType cur,ElemType &pre\_e);//获得e元素的前驱

status NextElem(SqList L,ElemType cur,ElemType &next\_e);//获得e元素的后继

status ListInsert(SqList &L,int i,ElemType e);//插入e元素在第i个元素之前

status ListDelete(SqList &L,int i,ElemType &e);//删除第i个元素

status ListTrabverse(SqList L); //遍历线性表并输出

status SaveList(SqList L,char FileName[]);//将线性表的元素写入文件中

status LoadList(SqList &L,char FileName[]);//将文件中的数据元素写入线性表

status AddList(LISTS &Lists,char ListName[]);//添加一个线性表表

status RemoveList(LISTS &Lists,char ListName[]);//删除一个指定名称的线性表

int LocateList(LISTS Lists,char ListName[]);//定位一个指定名称的线性表

status UpList(SqList &L);//升序排列

status DownList(SqList &L);//降序排列

status SubList(SqList L,SqList s);//判断s是不是L的子集

status InterList(SqList L,SqList S,SqList &p);//求两线性表的交集

status SumList(SqList L,SqList S,SqList &p);//就两线性表的并集

status DifferList(SqList L,SqList S,SqList &p);//求两线性表的差集

/\*--------------------------------------------\*/

int main(){

SqList L; int op=1;

L.elem=NULL;

int i,flag;

ElemType y,e,pre,next;

char name[30];

char filename[100],listname[100];

printf(" Menu for Linear Table On Sequence Structure \n");

printf("-------------------------------------------------\n");

printf(" 1. InitList 2. DestroyList\n");

printf(" 3. ClearList 4. ListEmpty\n");

printf(" 5. ListLength 6. GetElem\n");

printf(" 7. LocateElem 8. PriorElem\n");

printf(" 9. NextElem 10. ListInsert\n");

printf(" 11. ListDelete 12. ListTrabverse\n");

printf(" 13.SaveList 14.LoadList\n");

printf(" 15.AddList 16.RemoveList\n");

printf(" 17.LocateList 18.UpList\n");

printf(" 19.DownList 20.SubList\n");

printf(" 21.InterList 22.SumList\n");

printf(" 23.DifferList 0.exit\n");

printf("-------------------------------------------------\n");//菜单

while(op){

//system("cls"); printf("\n\n");

printf("请选择你的操作[0~23]:");

scanf("%d",&op);

switch(op){

case 1:

//printf("\n----IntiList功能待实现！\n");

if(InitList(L)==OK) printf("线性表创建成功！\n");

else printf("线性表已存在，创建失败！\n");

getchar();getchar();

break;

case 2:

//printf("\n----DestroyList功能待实现！\n");

if(DestroyList(L)==OK) printf("线性表销毁成功！\n");

else printf("线性表本身不存在，销毁失败！\n");

getchar();getchar();

break;

case 3:

//printf("\n----ClearList功能待实现！\n");

if(ClearList(L)==OK) printf("线性表已清空!\n");

else printf("线性表不存在!\n");

getchar();getchar();

break;

case 4:

//printf("\n----ListEmpty功能待实现！\n");

if(ListEmpty(L)==TRUE) printf("线性表为空！\n");

else if(ListEmpty(L)==FALSE) printf("线性表非空！\n");

else printf("线性表不存在！\n");

getchar();getchar();

break;

case 5:

//printf("\n----ListLength功能待实现！\n");

if(ListLength(L)==INFEASIBLE) printf("线性表不存在！\n");

else printf("线性表的长度为%d\n",ListLength(L));

getchar();getchar();

break;

case 6:

//printf("\n----GetElem功能待实现！\n");

if(!L.elem)

{

printf("线性表不存在！\n");

break;

}

printf("请输入获取元素的位序：");

scanf("%d",&i);

if(GetElem(L,i,e)==ERROR) printf("输入的%d值不合法！\n",i);

else printf("线性表第%d个值是%d!\n",i,e);

getchar();getchar();

break;

case 7:

//printf("\n----LocateElem功能待实现！\n");

if(!L.elem){

printf("不存在线性表！\n");

break;

}

if(LocateElem(L,e)==ERROR)

{

printf("线性表中不存在该元素!\n");

break;

}

printf("请输入要定位的元素：");

scanf("%d",&e);

printf("\n");

if(LocateElem(L,e)>0) printf("元素%d在线性表中的位置为%d!\n",e,LocateElem(L,e));

getchar();getchar();

break;

case 8:

//printf("\n----PriorElem功能待实现！\n");

if(!L.elem){

printf("线性表不存在！\n");

break;

}

printf("请输入元素以获得前驱：");

scanf("%d",&e);

printf("\n");

if(PriorElem(L,e,pre)==OK) printf("线性表中元素%d的前驱元素为%d\n",e,pre);

else if(PriorElem(L,e,pre)==ERROR) printf("没有前驱元素！\n");

getchar();getchar();

break;

case 9:

//printf("\n----NextElem功能待实现！\n");

if(!L.elem){

printf("线性表不存在！\n");

break;

}

printf("请输入元素以获得后继：");

scanf("%d",&e);

printf("\n");

if(NextElem(L,e,next)==OK) printf("线性表中元素%d的后继元素为%d\n",e,next);

else if(NextElem(L,e,next)==ERROR) printf("没有后继元素！\n");

getchar();getchar();

break;

case 10:

//printf("\n----ListInsert功能待实现！\n");

if(!L.elem){

printf("线性表不存在！\n");

break;

}

printf("请输入要插入的位置及元素：");

scanf("%d",&i);scanf("%d",&e);

printf("\n");

if(ListInsert(L,i,e)==OK) printf("插入成功！\n");

else if(ListInsert(L,i,e)==ERROR) printf("输入的值%d不合法！\n",i);

getchar();getchar();

break;

case 11:

//printf("\n----ListDelete功能待实现！\n");

if(!L.elem){

printf("线性表不存在！\n");

break;

}

printf("请输入要删除元素的位序：");

scanf("%d",&i);

printf("\n");

if(ListDelete(L,i,e)==OK) printf("删除成功，删除元素为%d!\n",e);

else if(ListDelete(L,i,e)==ERROR) printf("删除失败！输入值不合法!\n");

getchar();getchar();

break;

case 12:

//printf("\n----ListTrabverse功能待实现！\n");

if(ListTrabverse(L)==INFEASIBLE) printf("线性表不存在！\n");

else if(L.length==0) printf("线性表是空表！\n");

getchar();getchar();

break;

case 13:

//SaveList函数

if(!L.elem){

printf("线性表不存在！\n");

break;

}

printf("请输入文件名：");

scanf("%s",filename);

if(SaveList(L,filename)==OK) printf("\n写入文件成功！\n");

else if(SaveList(L,filename)==ERROR) printf("写入文件失败！\n");

getchar();getchar();

break;

case 14:

//LoadList函数

if(!L.elem){

printf("线性表不存在！\n");

break;

}

printf("请输入文件名：");

scanf("%s",filename);

if(LoadList(L,filename)==OK) printf("文件数据输入线性表成功！\n");

else if(LoadList(L,filename==ERROR)) printf("文件数据输入线性表失败！\n");

//else printf("线性表不存在！\n");

getchar();getchar();

break;

case 15:

//AddList函数

printf("请输入要添加线性表的个数：\n");

int n;

scanf("%d",&n);

while(n--){

printf("请输入要增加的线性表名称：\n");

scanf("%s",name);

AddList(Lists,name);

}

printf("线性表个数为%d\n",Lists.length);

printf("下面给出多个线性表的遍历：\n");

for(n=0;n<Lists.length;n++)

{

printf("%s ",Lists.elem[n].name);

ListTrabverse(Lists.elem[n].L);

putchar('\n');

}

getchar();getchar();

break;

case 16:

//RemoveList函数

printf("请输入删除线性表的名称") ;

scanf("%s",listname);

if(RemoveList(Lists,listname)) printf("%s线性表删除成功!\n",listname);

else printf("删除失败！\n");

getchar();getchar();

break;

case 17:

//LocateList函数

printf("请输入所需查找的线性表名称");

scanf("%s",listname);

if(LocateList(Lists,listname)) printf("%s线性表的位置为%d!\n",listname,LocateList(Lists,listname));

else printf("不存在该线性表！\n");

break;

case 18:

if(UpList(L)==INFEASIBLE){

printf("线性表不存在！\n");

break;

}

printf("升序后的线性表为：\n");

ListTrabverse(L);

printf("\n");

break;

case 19:

if(DownList(L)==INFEASIBLE){

printf("线性表不存在!\n");

break;

}

printf("降序后的线性表为：\n");

ListTrabverse(L);

break;

case 20:

flag=SubList(Lists.elem[0].L,Lists.elem[1].L);

if(flag==INFEASIBLE){

printf("多线性表中不存在两个线性表!\n");

break;

}

if(flag==OK){

printf("%s是%s的一个子集！\n",Lists.elem[1].name,Lists.elem[0].name);

}

else{

flag=SubList(Lists.elem[1].L,Lists.elem[0].L);

if(flag==OK){

printf("%s是%s的一个子集！\n",Lists.elem[0].name,Lists.elem[1].name);

}

else{

printf("二者不存在子集关系！\n");

}

}

break;

case 21:

if(InterList(Lists.elem[0].L,Lists.elem[1].L,Lists.elem[2].L)==INFEASIBLE) printf("线性表不存在\n");

if(Lists.elem[2].L.length==0) printf("交集为空！");

else {

printf("两线性表的交集是：");

ListTrabverse(Lists.elem[2].L);

}

printf("\n");

break;

case 22:

if(SumList(Lists.elem[0].L,Lists.elem[1].L,Lists.elem[2].L)==INFEASIBLE) printf("线性表不存在\n");

if(Lists.elem[2].L.length==0) printf("并集为空！");

else {

printf("两线性表的并集是：");

ListTrabverse(Lists.elem[2].L);

}

printf("\n");

break;

case 23:

if(DifferList(Lists.elem[0].L,Lists.elem[1].L,Lists.elem[2].L)==INFEASIBLE) printf("线性表不存在\n");

if(Lists.elem[2].L.length==0) printf("差集为空！");

else {

printf("两线性表的差集是：");

ListTrabverse(Lists.elem[2].L);

}

printf("\n");

break;

case 0:

break;

}//end of switch

}//end of while

printf("欢迎下次再使用本系统！\n");

return 0;

}//end of main()

/\*--------page 23 on textbook --------------------\*/

status InitList(SqList &L)

// 线性表L不存在，构造一个空的线性表，返回OK，否则返回INFEASIBLE。

{

if(L.elem) return INFEASIBLE;//存在线性表，如果再构造线性表，会覆盖掉原先的元素，所以返回INFEASIBLE

L.elem=(ElemType\*)malloc(sizeof(ElemType)\*LIST\_INIT\_SIZE); //给线性表分配空间

if(!L.elem) return INFEASIBLE;//如果分配失败，返回INFEASIBLE

L.length=0;//构造空的线性表，表长为0

L.listsize=LIST\_INIT\_SIZE;//设定最大长度

return OK;

}

status DestroyList(SqList& L)

// 如果线性表L存在，销毁线性表L，释放数据元素的空间，返回OK，否则返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

L.length=0;//让线性表为空

free(L.elem);//释放数据元素的空间

L.elem=NULL;//L.elem指针仍旧存在，成为了野指针，不符合释放空间的要求，释放后使其指向空指针

return OK;

}

status ClearList(SqList& L)

// 如果线性表L存在，删除线性表L中的所有元素，返回OK，否则返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

L.length=0;//使线性表为空来删除所有元素

return OK;

}

status ListEmpty(SqList L)

// 如果线性表L存在，判断线性表L是否为空，空就返回TRUE，否则返回FALSE；如果线性表L不存在，返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

if(L.length==0) return TRUE;//线性表为空，返回1

else return FALSE;//线性表不为空，返回0

}

status ListLength(SqList L)

// 如果线性表L存在，返回线性表L的长度，否则返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

return L.length;//线性表存在，返回其长度值

}

status GetElem(SqList L,int i,ElemType &e)

// 如果线性表L存在，获取线性表L的第i个元素，保存在e中，返回OK；如果i不合法，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

if(i>L.length||i<=0) return ERROR;//i的值不合法，元素的位置无负数并且有最大值

e=L.elem[i-1];//i值合法时，返回第i个元素

return OK;

}

status LocateElem(SqList L,ElemType e)

// 如果线性表L存在，查找元素e在线性表L中的位置序号并返回OK；如果e不存在，返回ERROR；当线性表L不存在时，返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

int i;//用来表示元素在线性表中的逻辑位置

for(i=0;i<L.length;i++)//查找线性表中是否有元素与e相等

if(e==L.elem[i]) return i+1;//若有元素与e相等，直接返回逻辑位置的值

return ERROR;//没有与e相等的值，返回0

}

status PriorElem(SqList L,ElemType e,ElemType &pre)

// 如果线性表L存在，获取线性表L中元素e的前驱，保存在pre中，返回OK；如果没有前驱，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//如果线性表不存在，返回INFEASIBLE

if(e==L.elem[0]) return ERROR;//线性表中的第一个元素与e相等，但第一个元素没有直接前驱，返回ERROR

int i;//用来记录线性表元素的逻辑位置

for(i=1;i<L.length;i++)//查找有前驱的元素中是否有与e值相等的元素

if(e==L.elem[i]){//发现与e相等的元素

pre=L.elem[i-1];//记录前驱元素

return OK;

}

return ERROR;//未找到与e相等的元素

}

status NextElem(SqList L,ElemType e,ElemType &next)

// 如果线性表L存在，获取线性表L元素e的后继，保存在next中，返回OK；如果没有后继，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//如果线性表不存在，返回INFEASIBLE

int i;//i用来标记元素的逻辑位置

for(i=0;i<L.length-1;i++)//在有后驱的元素中查找是否有与e相等的元素 最后一个元素无后驱

if(e==L.elem[i]){//找到与e相等的元素

next=L.elem[i+1];//记录后驱元素

return OK;

}

return ERROR;//未找到有后驱且与e相等的元素

}

status ListInsert(SqList &L,int i,ElemType e)

// 如果线性表L存在，将元素e插入到线性表L的第i个元素之前，返回OK；当插入位置不正确时，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(i<=0||i>L.length+1) return ERROR;//i的值不合法，无法插入

if(!L.elem) return INFEASIBLE;//线性表不存在，return INFEASIBLE

if(L.length>=L.listsize){//线性表已满

L.elem=(ElemType\*)realloc(L.elem,(L.listsize+LISTINCREMENT)\*(sizeof(ElemType)));//对线性表重新分配空间，增加线性表的空间

if(!L.elem) return OVERFLOW;//重新分配空间失败

L.listsize+=LISTINCREMENT;//线性表的最大表长增加

}

int j;

for(j=L.length;j>=i;j--)//从i个元素到最后一个元素都后移一位

L.elem[j]=L.elem[j-1];

L.elem[i-1]=e;//插入e到i前面，即变成第i位

L.length++;//线性表的表长增加

return OK;

}

status ListDelete(SqList &L,int i,ElemType &e)

// 如果线性表L存在，删除线性表L的第i个元素，并保存在e中，返回OK；当删除位置不正确时，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

if(i>L.length||i<=0) return ERROR;//i值不合法，线性表不存在第i个元素

e=L.elem[i-1];//记录第i个元素

int j;

for(j=i-1;j<L.length-1;j++)//从第i+1个元素 到最后一个元素，前移一位

L.elem[j]=L.elem[j+1];

L.length--;//线性表的表长减一

return OK;

}

status ListTrabverse(SqList L)

// 如果线性表L存在，依次显示线性表中的元素，每个元素间空一格，返回OK；如果线性表L不存在，返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//线性表不存在，返回INFEASIBLE

int i;

if(L.length>0){//线性表非空

for(i=0;i<L.length-1;i++)

printf("%d ",L.elem[i]);

printf("%d",L.elem[i]);//逐个输出元素

}

return OK;

}

status SaveList(SqList L,char FileName[])

// 如果线性表L存在，将线性表L的的元素写到FileName文件中，返回OK，否则返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;//如果不存在线性表，返回INFEASIBLE

FILE \*file=fopen(FileName,"w");//以“写”的方式打开文件

if(!file) return ERROR;//打开文件失败

int i;

for(i=0;i<L.length;i++){//向文件中写入元素

fprintf(file,"%d ",L.elem[i]);//数字元素之间一定要有空格，不然从文件中读取的时候只能读出来一个数

}

fclose(file);//关闭文件

return OK;

}

status LoadList(SqList &L,char FileName[])

// 如果线性表L存在，将FileName文件中的数据读入到线性表L中，返回OK，否则返回INFEASIBLE。

{

if(!L.elem) return INFEASIBLE;

FILE \*file=fopen(FileName,"r");//以读的方式打开文件

if(!file) return ERROR;

ElemType e;//将读出的数据放入e所指的地方

while(fscanf(file,"%d",&e)!=EOF){//读取数据到文件末尾

if(L.length>=L.listsize){//如果文件的数据多于开辟的空间

L.elem=(ElemType\*)realloc(L.elem,(L.listsize+LISTINCREMENT)\*(sizeof(ElemType)));

if(!L.elem) return OVERFLOW;

L.listsize+=LISTINCREMENT;

}//重新分配空间

L.elem[L.length]=e;//向线性表写入数据

L.length++;//表长增加

}

fclose(file);//关闭文件

return OK;

}

status AddList(LISTS &Lists,char ListName[])

// 只需要在Lists中增加一个名称为ListName的线性表

{

int i=Lists.length,j=0;//用i记录Lists中的数组长度，简便

if(i>=10) return ERROR;

while((\*(Lists.elem[i].name+j)=ListName[j])!='\0') j++;//将名称复制到Lists的数组中总后一个元素的后面

InitList(Lists.elem[i].L);//对增加的线性表初始化

Lists.length++;//线性表组的长度增加

printf("\n是否需要插入元素，插入输入1，不插入输入0：");

int y,e;

scanf("%d",&y);

if(y){

printf("\n请输入要插入的数据元素，以0结束：");

scanf("%d",&e);

while (e)//输入元素

{

ListInsert(Lists.elem[Lists.length-1].L,Lists.elem[Lists.length-1].L.length+1,e);

scanf("%d",&e);

}

}

return OK;

}

status RemoveList(LISTS &Lists,char ListName[])

// Lists中删除一个名称为ListName的线性表

{

int i,j;

char \*l;//用来指向线性表的名称，简便

for(i=0;i<Lists.length;i++)//遍历查看是否有名称为ListName的

{

l=Lists.elem[i].name;

j=0;

while(\*(l+j)==ListName[j]&&\*(l+j)&&ListName[j]) j++;//检测是否名称一样

if(!(\*(l+j))&&!ListName[j]){//找到名称为ListName的线性表

for(j=i;j<Lists.length-1;j++)//删除该线性表

Lists.elem[j]=Lists.elem[j+1];//该线性表后面的线性表都前移一位

Lists.length--;//线性表组的数量减一

Lists.elem[j].L.elem=NULL;//对数组的最后一个元素进行释放

return OK;

}

}

return ERROR;//未找到名称为ListName的线性表

}

int LocateList(LISTS Lists,char ListName[])

// 在Lists中查找一个名称为ListName的线性表，成功返回逻辑序号，否则返回0

{

int i,j;

char \*l;

for(i=0;i<Lists.length;i++){

l=Lists.elem[i].name;//记录线性表的名称，更简便

j=0;

while(\*(l+j)==ListName[j]&&\*(l+j)&&ListName[j]) j++;//找与目标线性表名称相同的线性表

if(!\*(l+j)&&!ListName[j]){//找到指定名称的线性表

return i+1;//返回所找线性表的逻辑位置

}

}

return 0;//未找到指定线性表

}

//升序

status UpList(SqList &L)

{

if(!L.elem) return INFEASIBLE;

int i,j;

ElemType t;

for(i=0;i<L.length;i++)//排序

for(j=i+1;j<L.length;j++)

if(L.elem[i]>L.elem[j])

{

t=L.elem[i];

L.elem[i]=L.elem[j];

L.elem[j]=t;

}

return OK;

}

//降序

status DownList(SqList &L)

{

if(!L.elem) return INFEASIBLE;

int i,j;

ElemType t;

for(i=0;i<L.length;i++)

for(j=i+1;j<L.length;j++)

if(L.elem[i]<L.elem[j])

{

t=L.elem[i];

L.elem[i]=L.elem[j];

L.elem[j]=t;

}

return OK;

}

//判断子表,判断s是否为L 的子集

status SubList(SqList L,SqList s)

{

if(!s.elem||!L.elem) return INFEASIBLE;

int i,flag,j;

for(i=0;i<s.length;i++)

{

flag=0;//先假设该元素在另一个线性表中不存在

for(j=0;j<L.length;j++)

if(L.elem[j]==s.elem[i]){//判断子集

flag=1; //该元素存在于另一个线性表中

break;

}

if(!flag) return ERROR;//有元素不相等，则不是子集关系

}

return OK;

}

//求俩顺序表的交,L,S的交集保存在p中

int InterList(SqList L,SqList S,SqList &p)//求交集

{

if(!L.elem||!S.elem||!p.elem) return INFEASIBLE;

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

p.length=0;//清空储存表

while(i<L.length&&j<S.length){

while(L.elem[i]<S.elem[j]) i++;//过滤不相等的元素

while(L.elem[i]>S.elem[j]) j++;//过滤不相等的元素

if(i<L.length&&j<S.length&&L.elem[i]==S.elem[j]){

ListInsert(p,k+1,L.elem[i]);//相等的插入

k++;i++;j++;//记得此时需要改变i，j，不然会死循环

}

}

if(!k) return ERROR;//无交集

return OK;

}

//求两顺序线性表的并

status SumList(SqList L,SqList S,SqList &p)

{

if(!L.elem||!S.elem||!p.elem) return INFEASIBLE;

int i=0,j=0;

p.length=0;

while(i<L.length&&j<S.length)

{

while(i<L.length&&j<S.length&&L.elem[i]<S.elem[j]){

ListInsert(p,p.length+1,L.elem[i]);

i++;

}

while(j<S.length&&i<L.length&&L.elem[i]>S.elem[j])

{

ListInsert(p,p.length+1,S.elem[j]);

j++;

}

while(j<S.length&&i<L.length&&L.elem[i]==S.elem[j])

{

ListInsert(p,p.length+1,S.elem[j]);

j++;i++;

}

}

if(i<L.length)

while(i<L.length)

{

ListInsert(p,p.length+1,L.elem[i]);

i++;

}

if(j<S.length)

while(j<S.length){

ListInsert(p,p.length+1,S.elem[j]);

j++;

}

return OK;

}

status DifferList(SqList L,SqList S,SqList &p)

//求两线性表的差集

{

if(!L.elem||!S.elem||!p.elem) return INFEASIBLE;

int i=0,j=0;

p.length=0;

while(i<L.length&&j<S.length)

{

while(i<L.length&&j<S.length&&L.elem[i]<S.elem[j]){

ListInsert(p,p.length+1,L.elem[i]);

i++;

}

while(j<S.length&&i<L.length&&L.elem[i]>S.elem[j])

{

ListInsert(p,p.length+1,S.elem[j]);

j++;

}

while(j<S.length&&i<L.length&&L.elem[i]==S.elem[j])

{

j++;i++;

}

}

if(i<L.length)

while(i<L.length)

{

ListInsert(p,p.length+1,L.elem[i]);

i++;

}

if(j<S.length)

while(j<S.length){

ListInsert(p,p.length+1,S.elem[j]);

j++;

}

return OK;

}

**附录B 基于链式存储结构线性表实现的源程序**

#include<stdio.h>

#include<malloc.h>

#include<stdlib.h>

#include<string.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define OVERFLOW -2

typedef int status;

typedef int ElemType; //数据元素类型定义

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

typedef int ElemType;

typedef struct LNode{ //单链表（链式结构）结点的定义

ElemType data;

struct LNode \*next;

}LNode,\*LinkList;

typedef struct{ //线性表的集合类型定义

struct { char name[30];

LinkList L;

} elem[10];

int length;

}LISTS;

status InitList(LinkList &L);//初始化一个线性表

status DestroyList(LinkList &L);//销毁一个线性表

status ClearList(LinkList &L);//清空一个线性表

status ListEmpty(LinkList L);//对一个线性表判空

int ListLength(LinkList L);//求线性表的表长

status GetElem(LinkList L,int i,ElemType &e);//获取线性表的指定位置的元素

status LocateElem(LinkList L,ElemType e);//对指定元素在线性表中进行定位

status PriorElem(LinkList L,ElemType e,ElemType &pre);//获取指定元素的前驱

status NextElem(LinkList L,ElemType e,ElemType &next);//获取指定元素的后继

status ListInsert(LinkList &L,int i,ElemType e);//在指定位置插入一个元素

status ListDelete(LinkList &L,int i,ElemType &e);//删除指定位置的元素

status ListTraverse(LinkList L);//遍历线性表

status SaveList(LinkList L,char FileName[]);//将线性表数据写入文件

status LoadList(LinkList &L,char FileName[]);//将文件中的数据写入线性表

status AddList(LISTS &lists,char ListName[]);//向多线性表中插入线性表

status RemoveList(LISTS &lists,char ListName[]);//删除指定名称的线性表

int LocateList(LISTS lists,char ListName[]);//定位线性表的位置

status UpList(LinkList &L);//升序排列线性表

status DownList(LinkList &L);//降序排列线性表

status ReverseList(LinkList &L);//逆置线性表

status MergeList(LinkList &L,LinkList b);//合并两个顺序线性表，顺序为升序

LinkList L=NULL;

ElemType e,pre,next;

int i;

char filename[1000],listname[30];

LISTS lists;

int main()

{

int flag,op=1;

printf(" Menu for Linear Table On Sequence Structure \n");

printf("-------------------------------------------------\n");

printf(" 1. InitList 2. DestroyList\n");

printf(" 3. ClearList 4. ListEmpty\n");

printf(" 5. ListLength 6. GetElem\n");

printf(" 7. LocateElem 8. PriorElem\n");

printf(" 9. NextElem 10. ListInsert\n");

printf(" 11. ListDelete 12. ListTrabverse\n");

printf(" 13.SaveList 14.LoadList\n");

printf(" 15.AddList 16.RemoveList\n");

printf(" 17.LocateList 18.UpList\n");

printf(" 19.DownList 20.ReverseList\n");

printf(" 21.MergeList 0.exit\n");

printf("-------------------------------------------------\n");//菜单

while(op){

printf("请选择你的操作[0~21]:");

scanf("%d",&op);

switch(op){

case 1:

flag=InitList(L);

if(flag==OK) printf("线性表创建成功！\n");

else printf("线性表已存在，创建失败！\n");

getchar();getchar();

break;

case 2:

flag=DestroyList(L);

if(flag==OK) printf("线性表销毁成功！\n");

else printf("线性表本身不存在，销毁失败！\n");

getchar();getchar();

break;

case 3:

flag=ClearList(L);

if(flag==OK) printf("线性表已清空!\n");

else printf("线性表不存在!\n");

getchar();getchar();

break;

case 4:

flag=ListEmpty(L);

if(flag==TRUE) printf("线性表为空！\n");

else if(flag==FALSE) printf("线性表非空！\n");

else printf("线性表不存在！\n");

getchar();getchar();

break;

case 5:

flag=ListLength(L);

if(flag==INFEASIBLE) printf("线性表不存在！\n");

else printf("线性表的长度为%d\n",flag);

getchar();getchar();

break;

case 6:

if(!L)

{

printf("线性表不存在！\n");

break;

}

printf("请输入获取元素的位序：");

scanf("%d",&i);

flag=GetElem(L,i,e);

if(flag==ERROR) printf("输入的%d值不合法！\n",i);

else printf("线性表第%d个值是%d!\n",i,e);

getchar();getchar();

break;

case 7:

if(!L){

printf("不存在线性表！\n");

break;

}

printf("请输入要定位的元素：");

scanf("%d",&e);

flag=LocateElem(L,e);

printf("\n");

if(flag==ERROR) printf("线性表中不存在该元素!\n");

else printf("元素%d在线性表中的位置为%d!\n",e,flag);

getchar();getchar();

break;

case 8:

if(!L){

printf("线性表不存在！\n");

break;

}

printf("请输入元素以获得前驱：");

scanf("%d",&e);

printf("\n");

flag=PriorElem(L,e,pre);

if(flag==OK) printf("线性表中元素%d的前驱元素为%d\n",e,pre);

else if(flag==ERROR) printf("没有前驱元素！\n");

getchar();getchar();

break;

case 9:

if(!L){

printf("线性表不存在！\n");

break;

}

printf("请输入元素以获得后继：");

scanf("%d",&e);

printf("\n");

flag=NextElem(L,e,next);

if(flag==OK) printf("线性表中元素%d的后继元素为%d\n",e,next);

else if(flag==ERROR) printf("没有后继元素！\n");

getchar();getchar();

break;

case 10:

if(!L){

printf("线性表不存在！\n");

break;

}

printf("请输入要插入的位置及元素：");

scanf("%d",&i);scanf("%d",&e);

printf("\n");

flag=ListInsert(L,i,e);

if(flag==OK) printf("插入成功！\n");

else if(flag==ERROR) printf("输入的值%d不合法！\n",i);

getchar();getchar();

break;

case 11:

if(!L){

printf("线性表不存在！\n");

break;

}

printf("请输入要删除元素的位序：");

scanf("%d",&i);

printf("\n");

flag=ListDelete(L,i,e);

if(flag==OK) printf("删除成功，删除元素为%d!\n",e);

else if(flag==ERROR) printf("删除失败！输入值不合法!\n");

getchar();getchar();

break;

case 12:

flag=ListTraverse(L);

if(flag==INFEASIBLE) printf("线性表不存在！\n");

else if(L->next==NULL) printf("线性表是空表！\n");

getchar();getchar();

break;

case 13:

if(!L){

printf("线性表不存在！\n");

break;

}

printf("请输入文件名：");

scanf("%s",filename);

flag=SaveList(L,filename);

if(flag==OK) printf("\n写入文件成功！\n");

else if(flag==ERROR) printf("写入文件失败！\n");

getchar();getchar();

break;

case 14:

printf("请输入文件名：");

scanf("%s",filename);

flag=LoadList(L,filename);

if(flag==OK) printf("文件数据输入线性表成功！\n");

else if(flag==ERROR) printf("文件数据输入线性表失败！\n");

getchar();getchar();

break;

case 15:

//AddList函数

printf("请输入要添加线性表的个数：\n");

int n;

scanf("%d",&n);

while(n--){

printf("请输入要增加的线性表名称：\n");

scanf("%s",listname);

flag=AddList(lists,listname);

if(flag==OK) printf("线性表插入成功!\n");

else printf("线性表插入失败!\n");

}

printf("线性表个数为%d\n",lists.length);

printf("下面给出多个线性表的遍历：\n");

for(n=0;n<lists.length;n++)

{

printf("%s:",lists.elem[n].name);

flag=ListTraverse(lists.elem[n].L);

if(lists.elem[n].L->next==NULL) printf("该线性表为空表!");

putchar('\n');

}

getchar();getchar();

break;

case 16:

//RemoveList函数

printf("请输入删除线性表的名称：") ;

scanf("%s",listname);

if(RemoveList(lists,listname)) printf("%s线性表删除成功!\n",listname);

else printf("删除失败！\n");

getchar();getchar();

break;

case 17:

//LocateList函数

printf("请输入所需查找的线性表名称:");

scanf("%s",listname);

flag=LocateList(lists,listname);

if(flag!=ERROR) printf("%s线性表的位置为%d!\n",listname,flag);

else printf("不存在该线性表！\n");

break;

case 18:

flag=UpList(L);

if(flag==INFEASIBLE){

printf("线性表不存在！\n");

break;

}

printf("升序后的线性表为：\n");

ListTraverse(L);

printf("\n");

break;

case 19:

flag=DownList(L);

if(flag==INFEASIBLE){

printf("线性表不存在!\n");

break;

}

printf("降序后的线性表为：\n");

ListTraverse(L);

printf("\n");

break;

case 20:

flag=ReverseList(L);

if(flag==INFEASIBLE){

printf("线性表不存在!\n");

break;

}

printf("逆置后的线性表为：");

ListTraverse(L);

printf("\n");

break;

case 21:

printf("请确保多线性表中有两个线性表再进行操作!\n");

flag=MergeList(lists.elem[0].L,lists.elem[1].L);

if(flag==INFEASIBLE){

printf("多线性表中不存在两个线性表!\n");

break;

}

UpList(lists.elem[0].L);

UpList(lists.elem[1].L);

printf("归并后的线性表为：");

ListTraverse(lists.elem[0].L);

printf("\n");

break;

case 0:

break;

}//end of switch

}//end of while

printf("欢迎下次再使用本系统！\n");

}

status InitList(LinkList &L)

// 线性表L不存在，构造一个空的线性表，返回OK，否则返回INFEASIBLE。

{

if(L) return INFEASIBLE;//如果线性表已存在，无法再初始化

L=(LinkList)malloc(sizeof(LNode));//分配头节点空间

L->next=NULL;//指针赋空

return OK;

}

status DestroyList(LinkList &L)

// 如果线性表L存在，销毁线性表L，释放数据元素的空间，返回OK，否则返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

LinkList a,b;

a=L,b=L->next;//a为当前要释放空间的结点，b为后继结点

while(a){//a到达末尾时完毕

free(a);//释放当前空间

a=b;//对下一结点进行操作

if(a)//a不为空时才有后继结点

b=a->next;

}

L=NULL;

return OK;

}

status ClearList(LinkList &L)

// 如果线性表L存在，删除线性表L中的所有元素，返回OK，否则返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

LinkList a,b;

a=L->next;//从首元结点开始删除，头结点不删除

if(!a) return OK;//链表本身为空，直接返回

b=a->next;//b记录后继节点

while(a){//a不为空时才删除

free(a);

a=b;//对下一个结点操作

if(a)//a不为空才有后继结点

b=a->next;

}

L->next=NULL;//对头结点的next赋空

return OK;

}

status ListEmpty(LinkList L)

// 如果线性表L存在，判断线性表L是否为空，空就返回TRUE，否则返回FALSE；如果线性表L不存在，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

if(L->next) return FALSE;//首元结点为空，链表为空

else return TRUE;

}

int ListLength(LinkList L)

// 如果线性表L存在，返回线性表L的长度，否则返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

LinkList a;

a=L->next;//a为当前结点

int len=0;//开始还没计数时赋为0

while(a){//a不为空时才记录当前结点为一个元素

len++;//当前结点有元素，长度加一

a=a->next;//判断下一结点

}

return len;

}

status GetElem(LinkList L,int i,ElemType &e)

// 如果线性表L存在，获取线性表L的第i个元素，保存在e中，返回OK；如果i不合法，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

if(i<=0) return ERROR;

LinkList a;

int len=0;//当前所在位置的元素位置

a=L->next;

while(a){//a不为空时才计数

len++;//计数当前元素的位置

if(len==i){//找到要求的位置

e=a->data;

return OK;

}

a=a->next;

}

return ERROR;

}

status LocateElem(LinkList L,ElemType e)

// 如果线性表L存在，查找元素e在线性表L中的位置序号；如果e不存在，返回ERROR；当线性表L不存在时，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

int i=0;

LinkList a=L->next;

while(a){

i++;//记录当前的元素的位置序号

if(a->data==e) return i;//找到元素返回位置

a=a->next;

}

return ERROR;

}

status PriorElem(LinkList L,ElemType e,ElemType &pre)

// 如果线性表L存在，获取线性表L中元素e的前驱，保存在pre中，返回OK；如果没有前驱，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

if(!L->next) return ERROR;

LinkList a,b;

a=L->next;//a为首元结点

if(!a->next) return ERROR;//首元结点无前驱，若首元结点之后为空，就没有含有前驱的结点

b=a->next;//第一个有前驱的结点

while(b){

if(b->data==e){//找到指定元素

pre=a->data;//获取前驱元素

return OK;

}

a=b;b=a->next;//当前结点与前驱结点均后移

}

return ERROR;

}

status NextElem(LinkList L,ElemType e,ElemType &next)

// 如果线性表L存在，获取线性表L元素e的后继，保存在next中，返回OK；如果没有后继，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

if(!L->next) return ERROR;//线性表表为空

LinkList a=L->next;//第一个具有后继的结点

while(a->next){//最后一个具有后继的结点为后继不为空的结点

if(a->data==e){

next=a->next->data;//记录后继元素

return OK;

}

a=a->next;

}

return ERROR;

}

status ListInsert(LinkList &L,int i,ElemType e)

// 如果线性表L存在，将元素e插入到线性表L的第i个元素之前，返回OK；当插入位置不正确时，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

if(i<=0) return ERROR;//i值不合法

LinkList a=L;

int len=0;//当前结点的位置序号

while(a){//插入到第i个元素之前也就是插入到第i-1个元素之后，一直遍历到最后一个元素，最多插入到最后一个元素之后

if(len==i-1){//找到第i-1个元素，在该元素之后插入元素

LinkList b;//为插入的结点分配空间和位置

b=(LinkList)malloc(sizeof(LNode));

b->data=e;//插入元素赋值

b->next=a->next;

a->next=b;//插入b在a之后

return OK;

}

a=a->next;len++;

}

return ERROR;//遍历完毕，i值不合法

}

status ListDelete(LinkList &L,int i,ElemType &e)

// 如果线性表L存在，删除线性表L的第i个元素，并保存在e中，返回OK；当删除位置不正确时，返回ERROR；如果线性表L不存在，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

if(i<=0) return ERROR;//i值不合法

LinkList a=L,b;

int len=0;//当前到达的结点的位置序号

while(a){//删除第i个元素，也就是让第i-1个元素的next记录第i+1个元素

if(len==i-1){//到达第i-1个元素的位置

b=a->next;//记录要删除的结点，第i个元素

if(!b) return ERROR;//要删除的结点为空，i值不合法

e=b->data;//记录要删除的元素

a->next=b->next;//删除结点

free(b);//释放该节点的空间

b=NULL;//野指针赋空

return OK;

}

len++;a=a->next;

}

return ERROR;

}

status ListTraverse(LinkList L)

// 如果线性表L存在，依次显示线性表中的元素，每个元素间空一格，返回OK；如果线性表L不存在，返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

LinkList a=L->next;

while(a){

printf("%d",a->data);

a=a->next;

if(a) printf(" ");//最后一个元素后面不输出空格

}

return OK;

}

status SaveList(LinkList L,char FileName[])

// 如果线性表L存在，将线性表L的的元素写到FileName文件中，返回OK，否则返回INFEASIBLE。

{

if(!L) return INFEASIBLE;

LinkList a=L->next;

FILE \*file=fopen(FileName,"w");//以“写”的方式打开文件

if(!file) return ERROR;//打开文件失败

while(a){

fprintf(file,"%d ",a->data);

a=a->next;

}

fclose(file);//关闭文件!!!

return OK;

}

status LoadList(LinkList &L,char FileName[])

// 将FileName文件中的数据读入到线性表L中，返回OK，否则返回INFEASIBLE。

{

if(L) ClearList(L);

else InitList(L);

LinkList a=L,b=NULL;

FILE \*file=fopen(FileName,"r");//以读的方式打开文件

if(!file) return ERROR;

ElemType e;//将读出的数据放入e所指的地方

while(fscanf(file,"%d",&e)!=EOF){//读取数据到文件末尾

b=(LinkList)malloc(sizeof(LNode));

b->next=NULL;

b->data=e;

a->next=b;a=b;b=NULL;//在线性表的末尾插入结点，a表示最后一个元素的结点

}

fclose(file);

return OK;

}

status AddList(LISTS &lists,char ListName[])

//向多线性表中添加一个指定名称的线性表

{

int i=lists.length,j=1,flag;

if(i>=10) return ERROR;//超过数组边界

strcpy(lists.elem[i].name,ListName);//名字赋值

InitList(lists.elem[i].L);//初始化线性表

printf("是否要向线性表中插入元素，若要插入元素输入1，否则输入0：");

scanf("%d",&flag);

if(flag){

printf("\n请输入要插入的元素并且以0结尾\n");

scanf("%d",&flag);

while(flag){

ListInsert(lists.elem[i].L,j,flag);//插入元素

j++;

scanf("%d",&flag);

}

}

lists.length++;//多线性表的长度增加

return OK;

}

status RemoveList(LISTS &lists,char ListName[])

{

int i=lists.length,j,k;

if(i==0) return ERROR;

for(j=0;j<i;j++){

if(strcmp(lists.elem[j].name,ListName)==0){//名字相同

for(k=j;k<i-1;k++) lists.elem[k]=lists.elem[k+1];//删除该线性表

DestroyList(lists.elem[k].L);

//lists.elem[k].L=NULL;//销毁该线性表

lists.length--;//多线性表减少一个线性表

return OK;

}

}

return ERROR;

}

int LocateList(LISTS lists,char ListName[])

{

int i;

for(i=0;i<lists.length;i++)//i+1为当前到达的线性表的位置序号

{

if(strcmp(lists.elem[i].name,ListName)==0) return i+1;//找到指定线性表

}

return ERROR;

}

status UpList(LinkList &L)

{

if(!L)

return INFEASIBLE;

LinkList p,q,s;

int temp;

for(p=L->next;p->next!=NULL;p=p->next)

{

s=p;

for(q=p->next;q;q=q->next)//找到当前元素之后的最小值

{

if(q->data<s->data)

s=q;

}

if(s!=p)//数据少，交换数据域

{

temp=p->data;

p->data=s->data;

s->data=temp;

}

}

return OK;

}

status DownList(LinkList &L)

{

if(!L) return INFEASIBLE;

UpList(L);//升序

ReverseList(L);//逆置

return OK;

}

status ReverseList(LinkList &L)

{

if(!L)

return INFEASIBLE;

LinkList pre,now,next;

now=L->next;//now记录当前到达的结点的位置

pre=NULL;//pre记录逆置以后now该指向的结点

while(now){//一直到最后一个结点也逆置成功

next=now->next;//记录当前结点的下一结点

now->next=pre;//更改当前节点的下一结点，逆置

pre=now;

now=next;//当前结点与前驱结点均后移 ，对下一个结点进行操作

}

L->next=pre;//逆置结束后，pre指向最后一个元素结点，逆置后应该为首元结点

return OK;

}

status MergeList(LinkList &L,LinkList L2)

//将两个升序线性表归并为一个线性表，并且删除两个线性表中相同的值

{

if(!L||!L2)

return INFEASIBLE;

LinkList a=L->next,pre=L,temp,b=L2->next;

while(a&&b)//未到达两线性表结尾

{

while(a!=NULL&&a->data<b->data){//a的元素小，一直向后走

pre=a;a=pre->next;

}//a元素更大以后才准备插入b元素在a之前

while(a!=NULL&&b!=NULL&&a->data==b->data){//两元素相等，不插入

b=b->next;

}

if(a&&b&&a->data>b->data){//避免跳出第一个循环是相等的情况

pre->next=b;temp=b->next;b->next=a;pre=b;//将b元素插入在a之前，也就是pre之后

b=temp;//temp记录了第二个线性表中的下一个元素，并对b进行更新

}

}

if(!a) pre->next=b;//因为a达到结尾而跳出了循环，所以b有可能没插入完，此时b元素一定是大的，将第一个表的结尾指向b即可，最后一个结点时pre

return OK;

}