# HW1 线性表 实验报告

姓名：徐恒 学号：2050035 日期：2022 年 4 月 1 日

## 1.涉及数据结构和相关背景

顺序表是指采用顺序存储结构的线性表，它利用内存中的一片连续存储区域存放表中的所有元素。可以根据需要对表中的所有数据进行访问，元素的插入和删除可以在表中的任何位置进行。顺序表的基本操作，包括顺序表的创建，第 i 个位置插入一个新的元素、删除第 i 个元素、查找某元素、顺序表的销毁。

## 2. 实验内容

### 2.1 学生信息管理

#### 2.1.1 问题描述

定义一个包含学生信息（学号，姓名）的顺序表，使其具有如下功能：(1)根据指定学生个数，逐个输入学生信息；(2)给定一个学生信息，插入到表中指定的位置；(3)删除指定位置的学生记录；(4)分别根据姓名和学号进行查找，返回此学生的信息；(5) 统计表中学生个数。

#### 2.1.2 基本要求

(1) 根据指定学生个数，逐个输入学生信息；
(2) 给定一个学生信息，插入到表中指定的位置；
(3) 删除指定位置的学生记录；
(4) 分别根据姓名和学号进行查找，返回此学生的信息；
(5) 统计表中学生个数

#### 2.1.3 数据结构设计

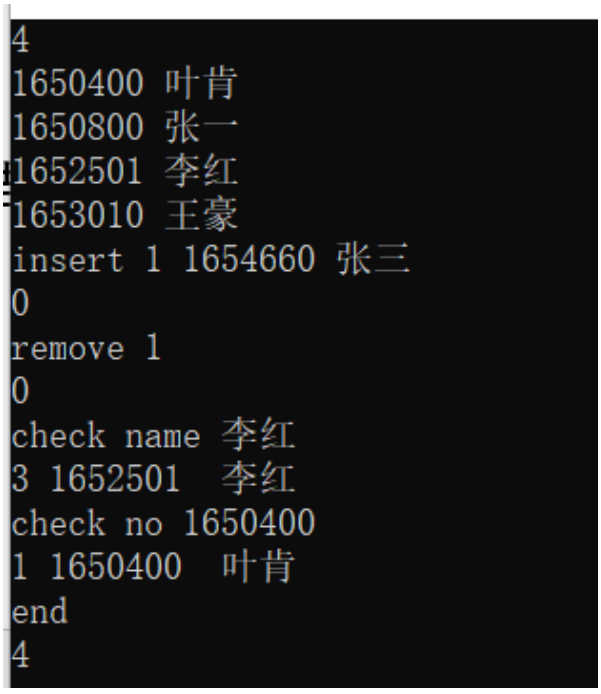
先输入学生个数，然后建立顺序表并输入元素，借助 for 循环输入要执行的操作函数，并进行函数的运行。其中，输入 insert 执行插入函数，要输入插入学生位置与插入的学生信息，合法返回 0，不然返回-1；输入 remove 执行删除函数，要输入删除学生位置，合法返回 0，不然返回-1；输入 check name 或 check no 执行查找函数，输入查找的学生姓名或学号，合法输出该学生信息，不然返回-1；输入 end 则输出此时表中总人数并终止程序。

#### 2.1.4 功能说明（函数、类）

struct person

```
{     char num[15];
      char name[15];}     //定义结构类型 person，用以存储学生信息
typedef struct{
      person *elem;
      int length;
      int listsize;
}SqList;   //定义结构类型 SqList，作为链表
int InitList_Sq(SqList &L) //L 为待进行操作的顺序表，该函数用于对 L 初始化
int ListInsert_Sq(SqList &L,int i,person e)//插入函数，在顺序表 L 的第 i 个位置插入元素 e,插入位置合法返回 0，不然返回-1
int ListDelete_Sq(SqList &L,int i,person &e)//删除函数，将顺序表 L 的第 i 个位置元素删除，并将其对应值传递给元素 e,删除位置合法返回 0，不然返回-1
int LocateElem_Sq2(SqList L,char *e)//查找学号函数，在顺序表 L 中查找第一个学号为 e 的元素，并将其位置序号返回，查找不到则返回-1
int LocateElem_Sq1(SqList L,char *e)//查找姓名函数，在顺序表 L 中查找第一个姓名为 e 的元素，并将其位置序号返回，查找不到则返回-1
```

### 2.1.5 调试分析（遇到的问题和解决方法）




出现数组越界的情况，需要加大空间。

### 2.1.6 总结和体会

在只循环一次的情况下，这题有两种解法
一种是用数组存储每次遍历的链表，这种比较简单但浪费一定空间。
第二种是快慢指针，快指针先移动 n 步，然后快慢指针一起移动直到快指针到表尾，此时慢指针的地址为所要删除节点的前一个节点。
注意：需要考虑边界条件，此题的边界条是当 n 为链表的长度时，此时删除的节点为头节点，需要返回 head->next。

## 2.2 小孩报数问题

### 2.2.1 问题描述

有 n 个人围坐一圈，给他们从 1 开始依次编号，现指定从第 s 个人开始报数，报到第 m 个人出列，然后从出列的下一个人开始重新报数，报到第 m 个人又出列，……如此反复，直到所有人全部出列。

### 2.2.2 基本要求

对于任意给定的 n,s,m，按出列次序依次输出所有人的编号，空格分割。

### 2.2.3 数据结构设计

依次输入 n,s,m，建立顺序表 L,对其初始化并输入其中元素,顺序表中第 i 个位置编号为 i+1。考虑第 s+m-1 位置的元素（即从 s 开始报数到的第 m 个人），若其超过总人数长度，则对编号取余数，找到报数人在顺序表中位置并输出与删除。循环此操作，直到顺序表为空。

### 2.2.4 功能说明（函数、类）

```
typedef struct{
    int *elem;
    int length;
}SqList; //  定义结构类型 SqList，作为顺序表
int InitList_Sq(SqList &L)   // L 为待进行操作的顺序表，该函数用于对 L 初始化
int ListDelete_sq(SqList &L,int i) //将顺序表 L 的第 i 个位置元素删除，若删除位置合法返回 0，不然返回-1
```

### 2.2.5 调试分析（遇到的问题和解决方法）

在确定所要删除的元素在顺序表中位置时，使用了循环 for(i=(s+m-2)%n;L.length>0;i=(i-1+m)%L.length)，但刚开始对 i 的初始值错误定义为了 s+m-1，调试中结果明显不对，后修改。

### 2.2.6 总结和体会

将基础的顺序表操作加以运用转化，在具体的问题分析中可以抽离出使用顺序表这一模型，并将求解的问题合理转化到对顺序表的删除，输出等一系列操作中，用顺序表解决实际问题。

## 2.3 顺序表的去重

### 2.3.1 问题描述

完成顺序表的去重运算，即将顺序表中所有重复的元素只保留第一个，其他均删除。

### 2.3.2 基本要求

创建一顺序表，对其去重后输出。

### 2.3.3 数据结构设计

输入建立顺序表的长度 n，对顺序表 L 初始化并输入其中元素。沿顺序表依次对每一个元素进行输出时，考虑之前以输出元素中是否有与其相等的，若有则跳过该元素，不然输出该元素并继续下一个。

### 2.3.4 功能说明（函数、类）

```
typedef struct{
    int *elem;
    int length;
}SqList;  //定义结构类型 SqList，作为顺序表
int InitList_Sq(SqList &L)  // L 为待进行操作的顺序表，该函数用于对 L 初始化
```

### 2.3.5 调试分析（遇到的问题和解决方法）

调试中并未出现错误。

### 2.3.6 总结和体会

利用顺序表的相关知识解决具体问题，本题解答重点在于输出时如何判断该元素是否已经输出过，即如何去重。

## 2.4 扑克牌游戏

### 2.4.1 问题描述

扑克牌有 4 种花色：黑桃（Spade）、红心（Heart）、梅花（Club）、方块（Diamond）。每种花色有 13 张牌，编号从小到大为：A，2，3，4，5，6，7，8，9，10，J，Q，K .现对一扑克牌堆，需定义添加、抽取、反转和弹出四种操作命令。

### 2.4.2 基本要求

1) 添加（Append）：添加一张扑克牌到牌堆的底部。如命令“Append Club Q”表示添加一张梅花 Q 到牌堆的底部。
2) 抽取（Extract）：从牌堆中抽取某种花色的所有牌，按照编号从小到大进行排序，并放到牌堆的顶部。如命令“Extract Heart”表示抽取所有红心牌，排序之后放到牌堆的顶部。
3)反转（Revert）：使整个牌堆逆序。
4)弹出（Pop）：如果牌堆非空，则除去牌堆顶部的第一张牌，并打印该牌的花色和数字；如果牌堆为空，则打印 NULL。

初始时牌堆为空。输入 n 个操作命令，执行对应指令。所有指令执行完毕后打印牌堆中所有牌花色和数字，如果牌堆为空，则打印 NULL。

### 2.4.3 数据结构设计

先建立空的链表 P 作为初始空扑克牌堆，输入操作数 n，依次输入执行的操作命令，最后输出执行完上述操作后的链表 P。

### 2.4.4 功能说明（函数、类）

```
struct poker
```

```
{     char col[8];
char num[3];} //定义结构类型 poker，用以存储扑克牌信息
typedef struct LNode{
      poker elem;
      struct LNode *next;
}LNode,*deck;  //定义链表结点以及指向其的指针
```

void Append(deck &P) //对链表 P 进行添加元素，即将要添加的元素插入在第一个结点处

void Extract(deck &P) //抽取链表 P 中某一花色的扑克，按编号从小到大排序后，整体插入在链表 P 第一个结点处

int Num(char num[]) //对 num 指向的字符变量进行量化处理，具体来说，“A”返回为数值 1，“2”返回数值 2，……，“10”返回数值 10，“J”返回数值 11，“Q”返回数值 12，“K”返回数值 13，其余返回数值-1.

void Revert(deck &P) //对链表 P 中的元素结点进行逆序处理，生成新的链表 P

void Pop(deck &P) //删除链表 P 中第一个元素结点，并输出该结点的元素信息

### 2.4.5 调试分析（遇到的问题和解决方法）

对题目给出的输入样例测试正确，但提交上出现多个 wrong 和 runtime error，经调试后发现 Extract 函数中排序用法错误导致逻辑出错。

### 2.4.6 总结和体会

编写时需要更仔细严谨一些，不然出错时再调整会耗费大量时间.

## 2.5 一元多项式的相加与相乘

### 2.5.1 问题描述

一元多项式是有序线性表的典型应用，用一个长度为 m 且每个元素有两个数据项（系数项和指数项）的线性表((p1,e1),(p2,e2),...,(pm,em))可以唯一地表示一个多项式。本题实现多项式的相加和相乘运算.

### 2.5.2 基本要求

输入两多项式，对其进行相加、相乘操作并输出。输入时不保证有序。

### 2.5.3 数据结构设计

输入多项式 Pa 的项数 m 并建立链表 Pa，输入多项式 Pb 的项数 n 并建立链表 Pb，输入操作指令 c，若为 0，则对多项式进行相加操作并打印；若为 1，则对多项式进行相乘操作并打印；若为 2，则对多项式进行相加与相乘操作并分别打印。

### 2.5.4 功能说明（函数、类）

```
struct term
{     float coef;
int expn;};  //定义结构类型 term，用以表示多项式的项
typedef struct LNode{
      term elem;
      struct LNode *next;
```

}LNode,*polynomial;  //定义链表结点以及指向其的指针
void CreatPolyn(polynomial &P,int m)  //创建一个长度为 m 的链表 P
void SortPolyn(polynomial &P) //对链表 P 中的项按指数从小到大排序
void PrintPolyn(polynomial P) //打印链表 P
void AddPolyn(polynomial &Pa,polynomial &Pb) //对多项式 Pa 和 Pb 相加，并将相加得到的多项式返回为 Pa
void MultiplyPolyn(polynomial &Pa,polynomial &Pb) //对多项式 Pa 和 Pb 相乘，并将相乘得到的多项式返回为 Pa

### 2.5.5 调试分析（遇到的问题和解决方法）

五个点出现 wrong，初步分析为输入超出缓冲区，未找到解决方案。

### 2.5.6 总结和体会

链接存储结构指的是用链表方法,值得注意的是，删除和插入较为灵活，不需要变动大多数元素，但是查找过程相对于数组这种顺序存储结构来说较为复杂，耗时巨大。在这里我们在一元多项式的表示与相加实验中，采用的是链式存储结构。

## 3. 实验总结

这一章的作业都是线性顺序表，线性表的顺序存储是指用一组地址连续的存储单元依次存储线性表中的各个元素，使得线性表在逻辑结构上相邻的元素存储在连续的物理存储单元中，即：通过数据元素物理存储的连续性来反应元素之间逻辑上的相邻关系。采用顺序存储结构存储的线性表通常简称为顺序表。

顺序存储的线性表的特点：线性表的逻辑顺序与物理顺序一致;数据元素之间的关系是以元素在计算机内“物理位置相邻”来体现。在数据表的第 i 个位置插入元素，在顺序表的第 i 个位置插入元素 e，首先将顺序表第 i 个位置的元素依次向后移动一个位置，然后将元素 e 插入第 i 个位置，移动元素要从后往前移动元素，即：先移动最后一个元素，在移动倒数第二个元素，依次类推；插入元素之前要判断插入的位置是否合法，顺序表是否已满，在插入元素之后要将表长 L->length++;顺序表的长度就是就顺序表中的元素的个数，由于在插入和删除操作中都有对数据表的长度进行修改，所以求表长只需返回 length 的值即可；查找顺序表中第 i 个元素的值（按序号查找），如果找到，将将该元素值赋给 e。查

找第 i 个元素的值时，首先要判断查找的序号是否合法，如果合法，返回第 i 个元素对应的值。删除表中的第 i 个元素 e，删除数据表中的第 i 个元素，需要将表中第 i 个元素之后的元素依次向前移动一位，将前面的元素覆盖掉。移动元素时要想将第 i+1 个元素移动到第 i 个位置，在将第 i+2 个元素移动 i+1 的位置，直到将最后一个元素移动到它的前一个位置，进行删除操作之前要判断顺序表是否为空，删除元素之后，将表长 L->length--