# 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。

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

## 1.1.1 实现基本操作

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

## 1.1.2 实现数据存储与读取

设计函数使得能够从文件中读取或者存储顺序存储线性表

## 1.1.3 实现多个线性表管理

能管理多个线性表

## 1.2 系统设计

## 1.2.1 程序总体框架

该系统包括用户交互模块，操作管理模块，多线性表管理模块，文件I/O模块，单个线性表操作模块。其中文件I/O模块包括写入与读取功能；单个线性表操作模块包括初始化表、销毁表、清空表、判定空表、求表长、获得元素、查找元素、获得前驱、获得后继、插入元素、删除元素、遍历表12个功能。

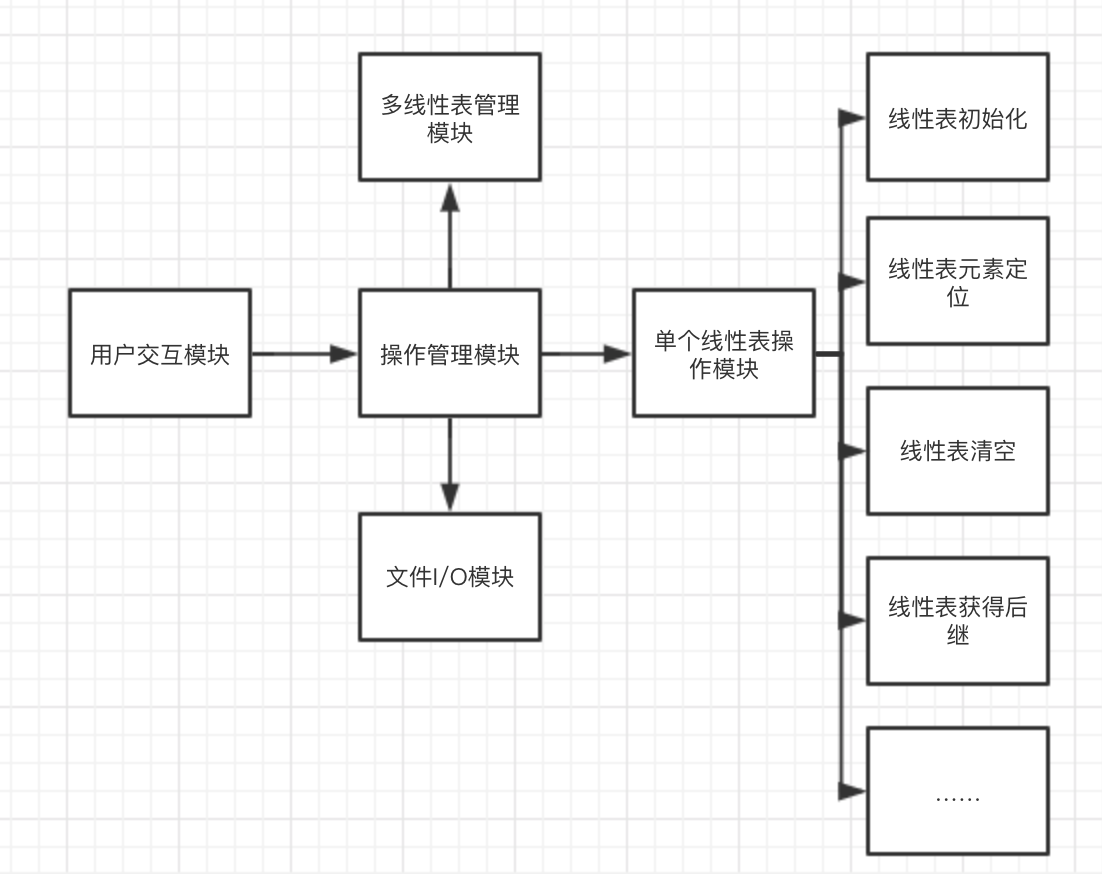


图1-1系统总体框架

## 1.2.2 系统环境

系统：macOS Catalina

IDE：clion

## 1.2.3 自定义部分

#define **TRUE** 1  
#define **FALSE** 0  
#define **OK** 1  
#define **ERROR** 0  
#define **INFEASTABLE** -1  
#define **OVERFLOW** -2  
#define **LIST\_INIT\_SIZE** 100  
#define **LISTINCREMENT** 10  
#define **LLIST\_SIZE** 10  
*//文件绝对路径***char** FILE\_NAME[100] = **"/Users/gikosei/Desktop/DS+/output"**;  
  
**typedef int** status;  
**typedef int** ElemType; *//数据元素类型定义***typedef struct** { *//顺序表（顺序结构）的定义* ElemType \*elem;  
 **int** length;  
 **int** listsize;  
} SqList;

## 1.3 系统实现

## 1.3.1 菜单栏

流程图：

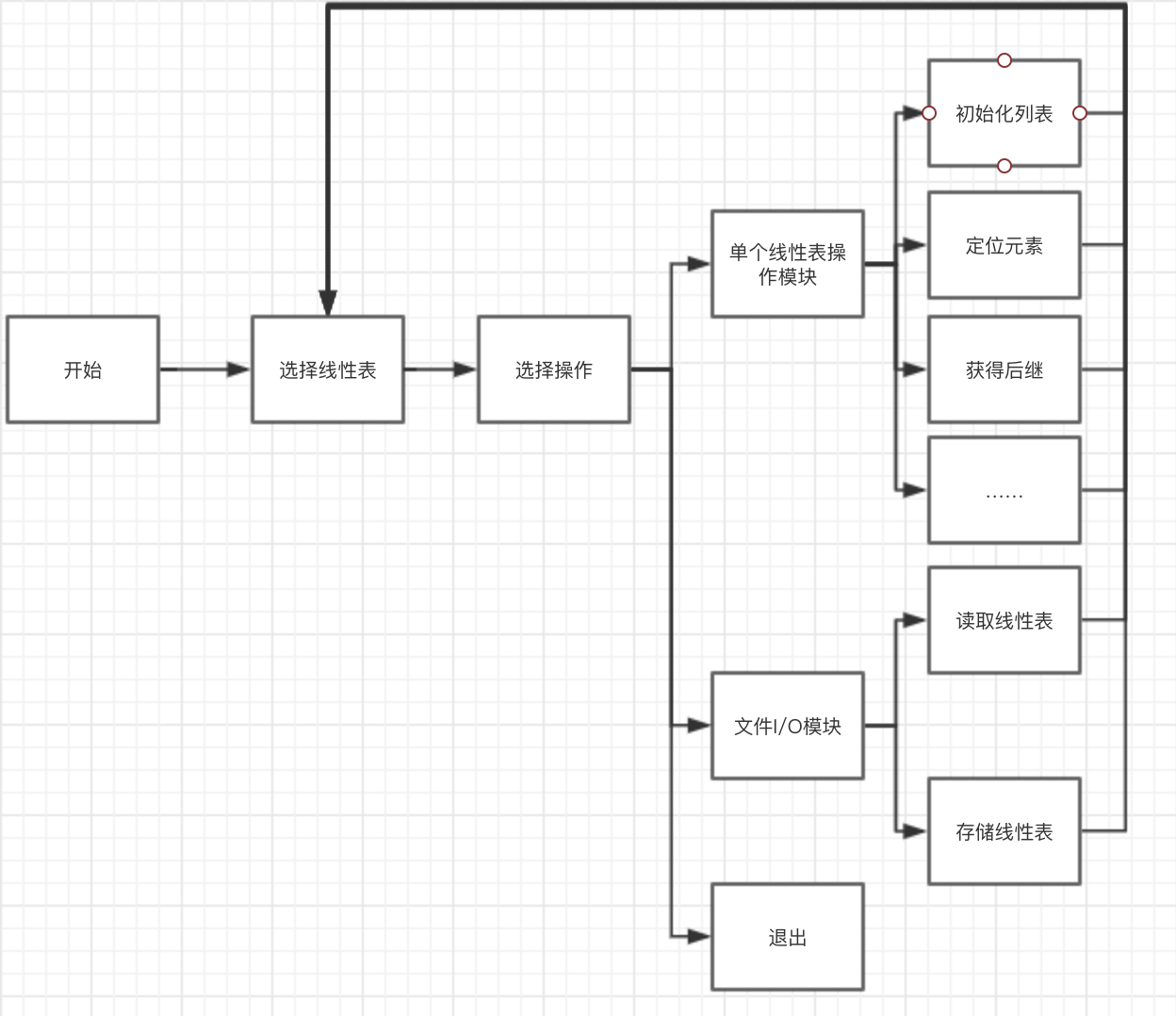


图1-2菜单栏流程图

## 1.3.2 函数实现与分析

⑴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。

时间复杂度：O(1)

意外情况：线性表分配内存失败，返回overflow

实现：分配内存，定义表长，表大小

⑵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。

时间复杂度：O(1)

实现：释放内存，表长，表大小都定义为0

⑶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。

时间复杂度：O(1)

意外情况：线性表内容不存在，返回error

实现：定义表长为0

⑷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。

时间复杂度：O(1)

实现：判断表长是否为0，是零返回TRUE，否则返回FALSE

⑸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。

时间复杂度：O(1)

实现：返回表长，若空表返回0

⑹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。

时间复杂度：O(n)

意外情况：1.线性内容不存在，返回error；2.i不在线性表范围内，返回error

实现：利用索引返回查找值

⑺查找元素：函数名称是LocateElem(L,e)；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相等的数据元素的位序，若这样的数据元素不存在，则返回值为0。

时间复杂度：O(n)

意外情况：1.线性内容不存在，返回error；2.线性表内无查找元素，返回0

实现：1.判断表空；2.遍历表

⑻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。

函数调用：调用LocateElem函数

时间复杂度：O(n)

意外情况：cur\_e不是线性表元素，或为第一个，返回error

流程图：

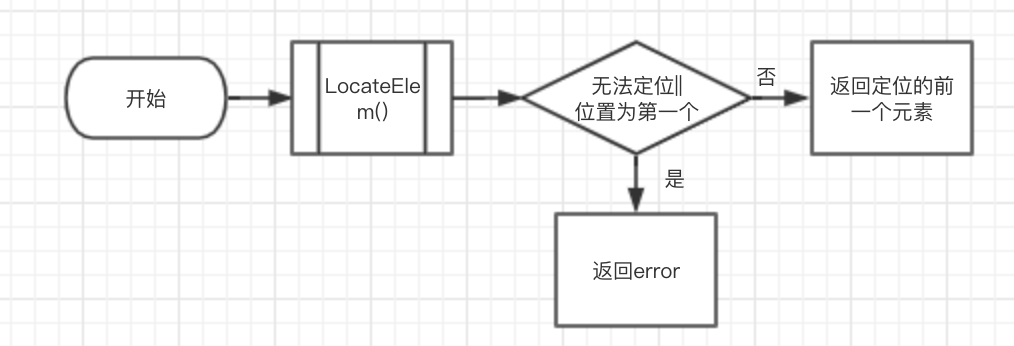


图1-3获得前驱流程图

⑼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。

函数调用：调用LocateElem函数

时间复杂度：O(n)

意外情况：cur\_e不是线性表元素，或为最后一个，返回error

实现：同获得前驱

⑽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。

时间复杂度：O(n)

意外情况：1.线性表无内容，返回error；2.线性表已满，扩充线性表；3.插入位置不在线性表范围内，返回error

流程图：

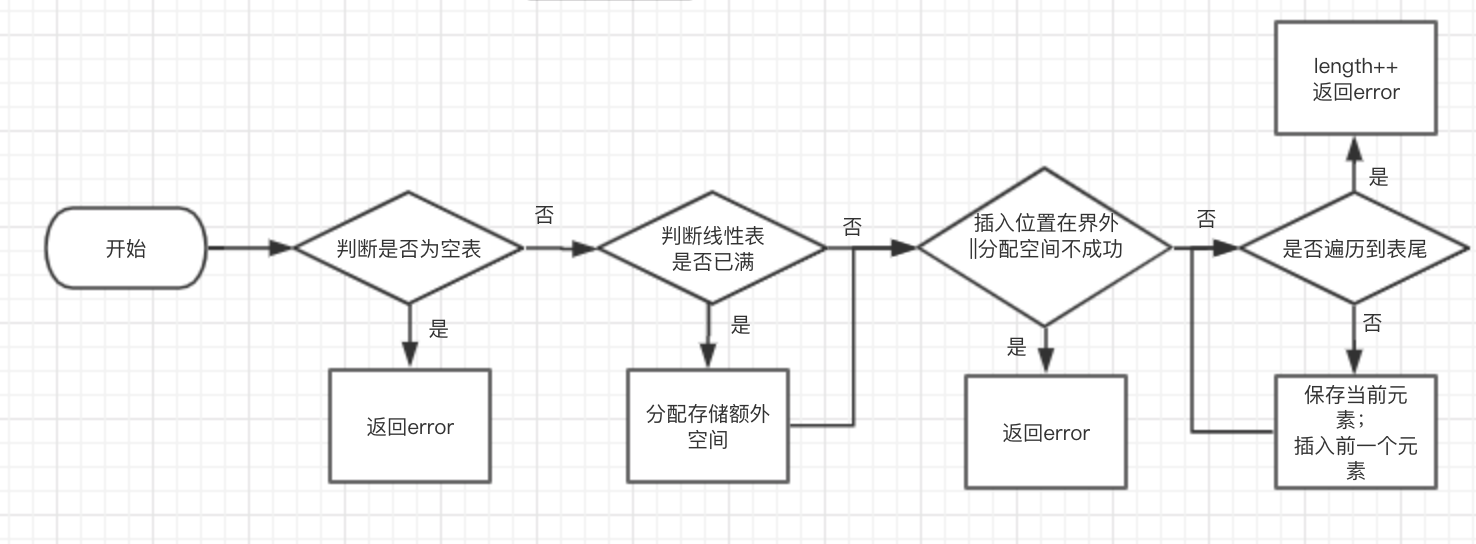


图1-4插入元素流程图

⑾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。

时间复杂度：O(n)

意外情况：1.线性表无内容，返回error；2.删除位置不在线性表范围内，返回error

实现：参考插入元素（不判断表是否已满）

⑿遍历表：函数名称是ListTraverse(L)，初始条件是线性表L已存在；操作结果是依次输出L的每个数据。

时间复杂度：O(n)

意外情况：线性表无内容，返回error

实现：输出分割线，遍历元素，输出分割线，返回表长

⒀存储表：f\_write(SqList l, char \*file\_name)，操作结果是将l的数据存到file\_name文件里。

时间复杂度：O(n)

意外情况：1.线性表无内容，返回error；2、文件未打开，输出File open error，返回error

实现：打开文件，判断文件是否打开，写入数据，关闭文件。

⒁读取表：f\_read(SqList &L, char \*file\_name)，操作结果是将file\_name的数据读入到L内。

时间复杂度：O(n)

意外情况：1.线性表无内容，返回error；2、文件未打开，输出File open error，返回error

实现：打开文件，判断文件是否打开，写入数据，同时每写入一个数据，表长加一，关闭文件。

**1.4 系统测试**

## 1.4.1 准备

定义函数createList初始化一个线性表（LList[0]）并填充元素，方便测试

status createList(SqList &L, int n) {  
 if (L.listsize < n) {  
 L.elem = (ElemType \*) realloc(L.elem, (L.listsize + LISTINCREMENT) \* sizeof(ElemType));  
 L.listsize += LISTINCREMENT;  
 }  
 for (int i = 0; i < n; i++) {  
 L.elem[i] = i;  
 }  
 L.length = n;  
}//测试用

## 1.4.2 测试

说明：

在选择操作前需要选择操作的线性表，所以实际上已经实现了对多个线性表进行管理，测试用函数的使用如下：

InitList(LList[0]);

createList(LList[0], 15);

（5）-（12）操作均为对于线性表LList[0]的操作，可以通过（12）的结果进行验证

菜单栏如下：

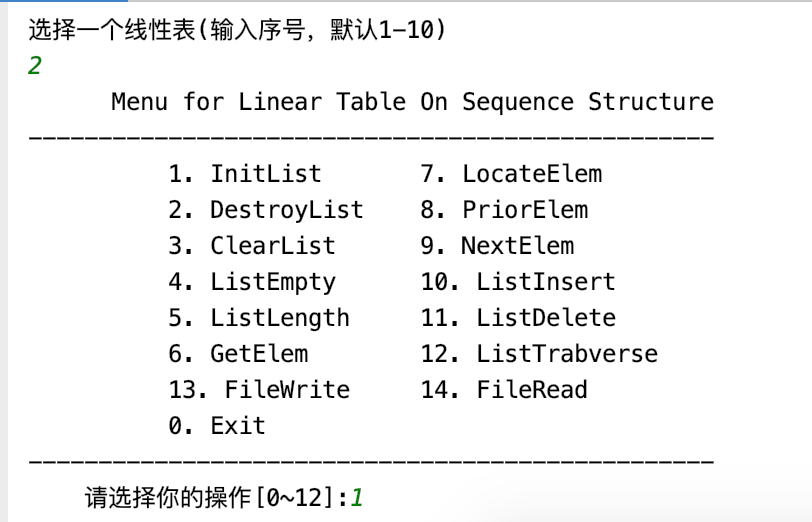


图1-5菜单栏

⑴初始化表

操作结果：

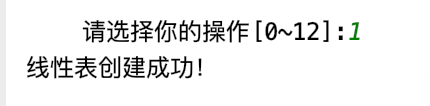


图1-6初始化表结果

⑵销毁表

操作结果：

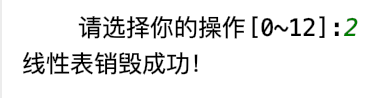


图1-7销毁表结果

⑶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。

操作结果：

表 1-1清空表结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 表内容存在 | 截屏2019-10-29下午7.56.44  图1-8-a清空表结果（内容存在） |
| 表内容不存在 | 截屏2019-10-29下午7.56.04  图1-8-b清空表结果内容不存在 |

⑷判定空表

操作结果：

表 1-2判定空表结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 表内容存在 | 返回长度 |
| 表内容不存在 | 返回0 |

⑸求表长

操作结果：

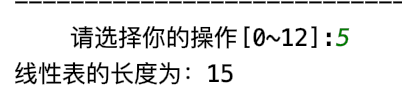


图1-9求表长结果

⑹获得元素

操作结果：

表 1-3获得元素结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.07.34  图1-10判定空表结果 |
| i不在线性表范围内 | 返回error，输出“位置有误” |
| 表内容不存在 | 返回error，输出“位置有误” |

⑺查找元素

操作结果：

表 1-4查找元素结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.11.13  图1-11查找元素结果 |
| i不在线性表范围内 | 返回error，输出“没有X元素” |
| 表内容不存在 | 返回error，输出“没有X元素” |

⑻获得前驱

操作结果：

表 1-5获得前驱结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.13.08  图1-12-a获得前驱结果（正常） |
| cur\_e不是线性表元素，或为第一个 | 截屏2019-10-29下午8.14.46  图1-12-b获得前驱结果（非正常） |

⑼获得后继

操作结果：

表 1-6获得后继结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.13.56  图1-13-a获得后继结果（正常） |
| cur\_e不是线性表元素，或为最后一个 | 截屏2019-10-29下午8.15.07  图1-13-b获得后继结果（非正常） |

⑽插入元素

操作结果：

表 1-7插入元素结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.17.44  图1-14插入元素结果 |
| 线性表无内容 | 输出“插入失败” |
| 线性表已满 | 扩充线性表 |
| 插入位置不在范围内 | 输出“插入失败” |

⑾删除元素

操作结果：

表 1-8删除元素结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.19.10  图1-15删除元素结果 |
| 线性表无内容 | 输出“插入失败” |
| 线性表已满 | 扩充线性表 |
| 插入位置不在范围内 | 输出“插入失败” |

⑿遍历表

操作结果：

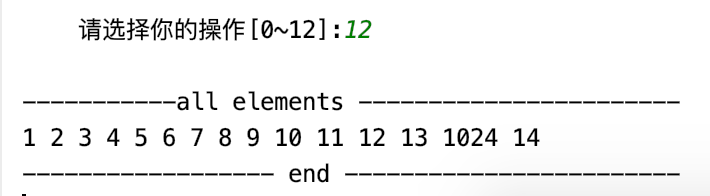


图1-16遍历表结果

⒀存储表

操作结果：

表 1-9存储表结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.25.16  图1-17存储表结果 |
| 线性表无内容 | 返回error，输出“写入文件失败” |
| 文件未打开 | 返回error，输出“open file error”  输出“写入文件失败” |

⒁读取表

注：需要先初始化线性表才能读取

操作结果：

表 1-10读取表结果

|  |  |
| --- | --- |
| 操作情况 | 操作结果 |
| 正常操作 | 截屏2019-10-29下午8.26.50  截屏2019-10-29下午8.27.26  图1-18读取表结果 |
| 线性表无内容 | 返回error，输出“读取文件失败” |
| 文件未打开 | 返回error，输出“open file error”  输出“写入文件失败” |

## 1.5 实验小结

1. 掌握了对顺序存储线性表的使用，加深了理解
2. 函数返回值为status可以方便判断函数返回的结果，便于逻辑处理
3. 将函数单独写入头文件方便处理，界面整洁
4. 注意函数传参传值和传引用的区别，传值为传入一份变量的拷贝，传参则是传入参数本身的指针，在需要修改参数本身，如插入，删除等操作时，需要传入引用
5. 注意边界条件的判断，如文件是否打开，指针是否为空，遍历条件等
6. 注意变量名称的规范性，要有具体含义