華中科技大學

课 程 实 验 报 告

课程名称： 数据结构实验

专业班级 CS2202

学　　号 U202215378

姓　　名 冯瑞琦

指导教师 李丹

报告日期 2022 年 5 月 1 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

1.1 问题描述

采用顺序表作为物理结构，构造一个具有菜单的功能演示系统。在主程序中显示完成函数调用所用的数字菜单，选择菜单数字调用相应的函数。操作时系统给出适当的输入提示，并显示调用结果。演示系统可进行多线性表管理。设计线性表文件保存和加载操作，可将生成的线性表保存到文件中，也可以从文件中读取线性表进行操作。

1.1.1 基本操作

1) InitList(L)

操作结果: 构造一个空的线性表 L。

2) DestroyList(L)

初始条件: 线性表 L 已存在。

操作结果: 销毁线性表 L。

3) ClearList(L)

初始条件: 线性表 L 已存在。

操作结果: 将 L 重置为空表。

4) ListEmpty(L)

初始条件: 线性表 L 已存在。

操作结果: 若 L 为空表，则返回 TRUE，否则返回 FALSE。

5) ListLength(L)

初始条件: 线性表 L 已存在。

操作结果: 返回 L 中数据元素个数。

6) GetElem(L,i,e)

初始条件: 线性表 L 已存在，1≤i≤ListLength(L)。

操作结果: 用 e 返回 L 中第 i 个数据元素的值。

7) LocateElem(L, e,compare())

初始条件: 线性表 L 已存在，compare() 是数据元素判定函数。

操作结果: 返回 L 中第 1 个与 e 满足关系 compare() 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) PriorElem(L,cur,pre)

初始条件: 线性表 L 已存在。

操作结果: 若 cur 是 L 的数据元素，且不是第一个，则用 pre 返回它的前驱，否则操作失败，pre 无定义。

9) NextElem(L,cur,next)

初始条件: 线性表 L 已存在。

操作结果: 若 cur 是 L 的数据元素，且不是最后一个，则用 next 返回它的后继，否则操作失败，nex 无定义。

10) ListInsert(L,i, e)

初始条件: 线性表 L 已存在，1≤i≤ListLength(L)+1。

操作结果: 在 L 中第 i 个位置之前插人新的数据元素 e，L 的长度加 1。

11) ListDelete(L,i,e)

初始条件: 线性表 L 已存在且非空，1≤≤ListLength(L)。

操作结果: 删除 L 的第主个数据元素，并用 · 返回其值，工的长度减 1。

12) ListTraverse(L,visit())

初始条件: 线性表 L 已存在。

操作结果: 依次对 L 的每个数据元素调用函数 viait(。一旦 visit() 失败，则操作失败。

1.2 系统设计

1.2.1 数据存储结构与形式

线性表的物理数据结构如下:

```
typedef struct{  //顺序表（顺序结构）的定义
        ElemType * elem;
```

```
        int length;
        int listsize;
}SqList;
```

若要实现多线性表管理，则定义一个线性表的结构数组，对线性表编号进行操作。

1.2.2 系统总体框架

通过 while 循环与菜单界面，用户通过选择菜单中的选项实现交互，使用 op 变量获取用户选择选项值 (op 初始化值为 1，以便第一次能进入循环)。

进入循环后系统首先显示功能菜单，然后用户输入选择 0-18, 其中 1-12 分别代表线性表的一个基本运算，13 与 14 选项分别是线性表的文件保存，以及显示全部线性表信息 (本选项用于方便查看在存在多个线性表的情况下，各个线性表的长度与名称信息)，15 是多线性表管理中选择操作的线性表的序号，16-18 是附加功能。在主函数中通过 switch 语句对应到相应的函数功能，执行完该功能后 break 跳出 switch 语句，继续执行 while 循环，直至用户输入 0 退出当前菜单系统。

系统总体框架如下图：

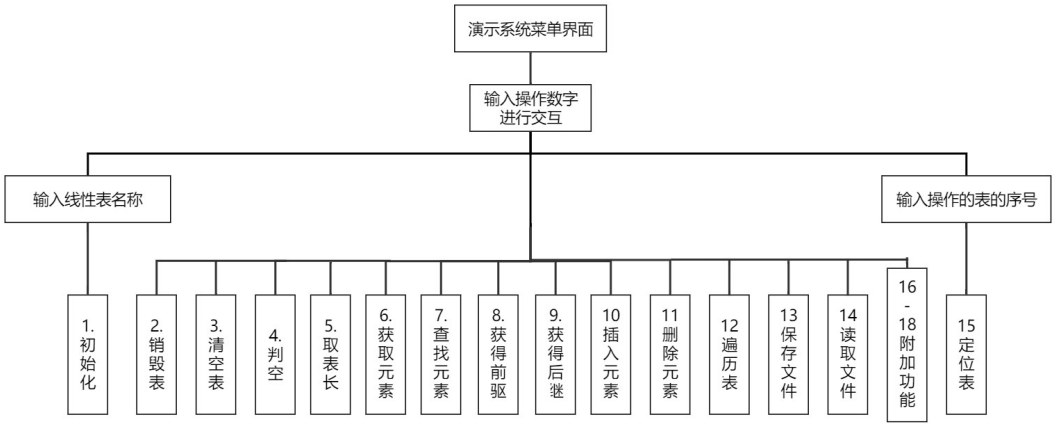

图 1-1 系统总体框架

1.3 系统实现

1.3.1 编程环境及运行环境描述

编程环境：采用 Dev-C++5.11 软件编写。

运行环境：微软 Windows 10 系统。

1.3.2 头文件及预定义常量说明

```
//头文件
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
//预定义常量
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE -1
#define OVERFLOW -2
#define MAX_NUM 10
#define LIST_INIT_SIZE 100
#define LISTINCREMENT  10
//数据元素类型定义
typedef int status;
typedef int ElemType;
```

1.3.3 算法设计与实现

1) 初始化表 status InitList(SqList L);

算法实现: 使用 malloc 函数分配连续内存空间，将首地址返回赋值给 L.elem, 将 L.length 初始化为 0，L.listsize 初始化为 LIST INIT SIZE。

2) 销毁表 status DestroyList(SqList L);

算法实现: 使用 free 函数释放掉以 L.elem 为首地址的连续内存空间，并将 L.length,L.listsize 重新赋值为 0。

3) 清空表 status ClearList(SqList L);

算法实现: 若线性表不存在返回 ERROR，若线性表存在，则将线性表的长度 L.length 的值置为 0。

4) 判空 status ListEmpty(SqList L);

算法实现: 若线性表不存在，返回-1, 若线性表存在则读取线性表 L.length, 若其值为 0 则返回 FALSE，否则返回 TRUE。

5) 求表长 int ListLength(SqList L);

算法实现: 若线性表不存在，返回-1, 若线性表存在则读取线性表 L.length, 若其值为 0 则返回 0，否则返回 L.length。

6) 获得元素 status GetElem(SqList L, int i,Elem Type e);

算法实现: 若 1≤i<L.Length 则通过随机访问数组的方式获取元素，将 L.elem[i-1] 的值赋值给 e；否则返回 ERROR。

7) 查找元素 status LocateElem(SqList L,ElemType e);

循环遍历线性表，将每一个元素与给定值比较，若相等就返回该元素序号，否则返回 ERROR。

8) 获得前驱 status PriorElem(SqList L,ElemType cur,ElemType pre);

从 i=1 开始遍历线性表，若 e=L.elem[i]，则将 L.elem[i-1] 赋值给 pre；若无元素符合，则返回 ERROR。

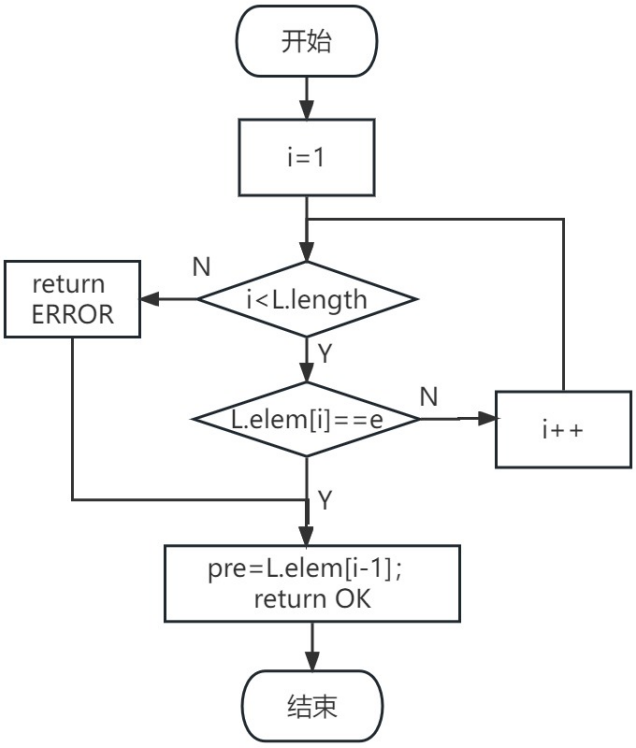

图 1-2　获得前驱

9) 获得后继 status NextElem(SqList L,ElemType cur,ElemType next);
遍历线性表，查找前 L.length-1 个元素是否符合，若符合则将其后继赋值给 next，若无元素符合，则返回 ERROR。

10) 插入元素 status ListInsert(SqList L,int i,ElemType e);
先判断输入 i 值是否合法（1<=i<=L.length+1），若不合法返回 ERROR；若存储空间已满，增加分配，先取插入位置的元素地址 q，将第 i 个元素保存在 e 中，然后将被删除元素位置及之后的元素前移，表长减一。

11) 删除元素 status ListDelete(SqList L,int i,ElemType e);
先判断输入 i 值是否合法（1<=i<=L.length+1），若不合法返回 ERROR；先取删除位置的元素地址 q，然后将插入位置及之后的元素后移，最后插入 e 元素到地址 q，表长加一。

12) 遍历表 status ListTraverse(SqList L);
若表长为 0，返回 0；否组遍历线性表，输出所有元素。

13) 保存到文件中 status SaveList(SqList L,char FileName[]);

用 fopen 打开文件，用 fprintf 将线性表的元素写到文件中，最后关闭文件。

14) 读取文件 status LoadList(SqList L,char FileName[]);

用 fopen 打开指定文件，用 fscanf 读取文件的数据，并将其赋值给线性表，同时线性表长度依次增加；若空间不足则用 realloc 重新分配空间，最后关闭文件。

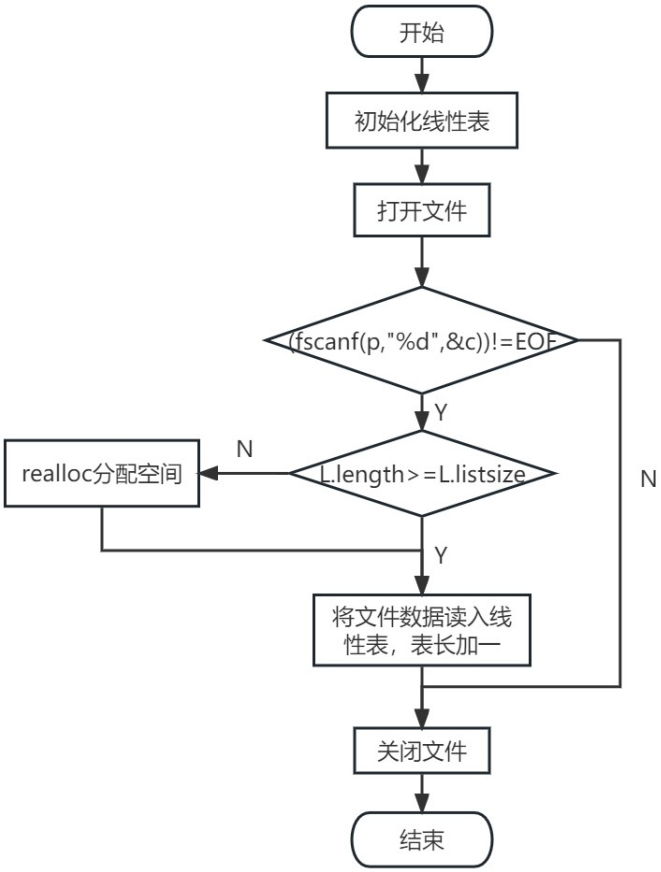

图 1-3　读取文件

15) 选择操作的线性表 LocateList

输入数字 i，在数组中第 i 个线性表上操作。

16) 最大连续字数和 status MaxSubArray(SqList L);

采用两层嵌套循环遍历线性表，求出各个连续子数和，并与当前最大值 max 比较，若大于 max 则将当前值赋给 max。

17) 和为 k 的连续子数组 status SubArrayNum(SqList L,ElemType k);

采用两层嵌套循环遍历线性表，求出各个连续子数和，并与所求值 k 比较，

若等于 k 则计数变量 count 加一，直到结束遍历，返回 count 的值。

18) 从小到大排序 status SortList(SqList L);

采用两层嵌套循环遍历线性表，比较相邻两元素大小，若后者小于前者，则利用中间变量 temp 将两者的值交换。

1.4 系统测试

当程序开始运行时，会进入菜单演示界面。

```
      Menu for Linear Table On Sequence Structure
------------------------------------------------
          1. InitList       11. ListDelete
          2. DertroyList    12. ListTraverse
          3. ClearList      13. SaveList
          4. ListEmpty      14. LoadList
          5. ListLength     15. LocateList
          6. GetElem        16. MaxSubArray
          7. LocateElem     17. SubArrayNum
          8. PriorElem      18. SortList
          9. NextElem
         10. ListInsert
------------------------------------------------
    请选择你的操作[0~18]:
```

图 1-4 菜单演示界面

输入数字 15，系统提示选择要操作的线性表的序号（如果不选择默认在 1 号线性表上操作）

```
    请选择你的操作[0~18]:15
请输入要在第几个表操作:
 *只支持在10个顺序表上操作*
```

图 1-5 选择要操作的线性表的序号

线性表的输入可以通过功能 1 初始化，同时命名和输入数据（图 1-6），也可通过功能 14 从文件中读入（图 1-7），读入时要先输入文件名。

```
    请选择你的操作[0~18]:1
请输入要创建的线性表名称:
A
线性表创建成功!
请输入元素，用空格隔开，以0结束!
```

图 1-6 初始化

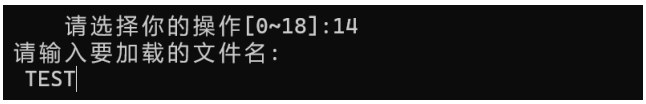

图 1-7　文件读入

接下来展示各功能的测试结果。

1) 初始化

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

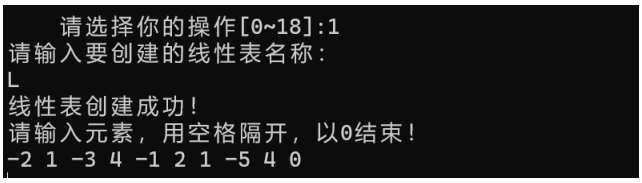

图 1-8　初始化

2) 销毁表

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

图 1-9　销毁表

3) 清空表

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

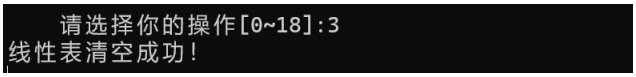

图 1-10　清空表

4) 判空

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

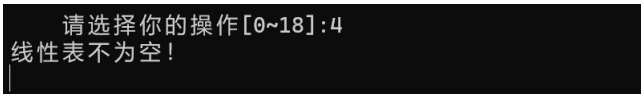

图 1-11　判空

测试用例：0（即空表），结果图如下。

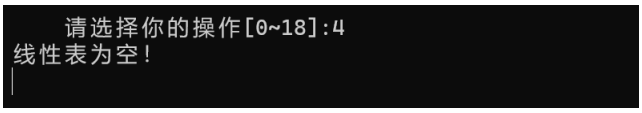

图 1-12　判空

测试用例：没有初始化直接调用功能 4，结果图如下。

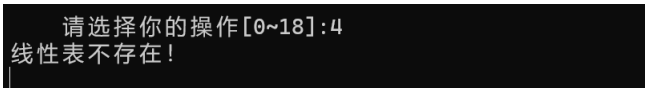

图 1-13　判空

5) 获取表长

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

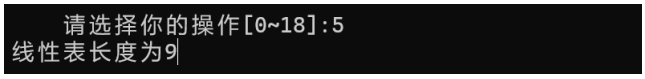

图 1-14　获取表长

测试用例：0（即空表），结果图如下。

图 1-15　获取表长

6) 获取元素

测试用例：-2 1 -3 4 -1 2 1 -5 4，获取第 2 个元素，结果图如下。

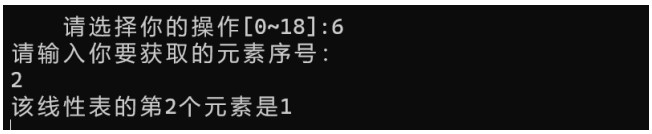

图 1-16　获取元素

测试用例：-2 1 -3 4 -1 2 1 -5 4，获取第 12 个元素，结果图如下。

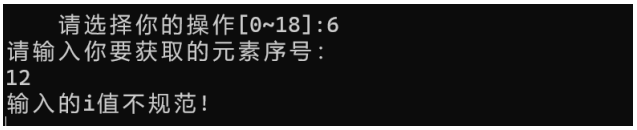

图 1-17　获取元素

7) 查找元素

测试用例：-2 1 -3 4 -1 2 1 -5 4，查找元素 4，结果图如下。

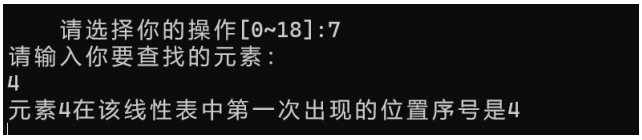

图 1-18　查找元素

测试用例：-2 1 -3 4 -1 2 1 -5 4，查找元素 18，结果图如下。

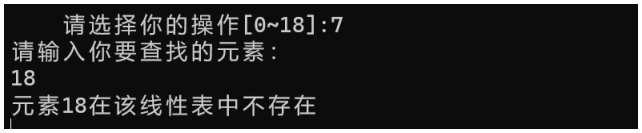

图 1-19　查找元素

8) 获取前驱

测试用例：-2 1 -3 4 -1 2 1 -5 4，获取-3 的前驱，结果图如下。

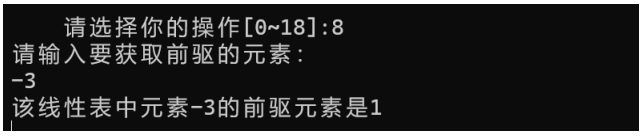

图 1-20　获取前驱

9) 获取后继

测试用例：-2 1 -3 4 -1 2 1 -5 4，获取-3 的后继，结果图如下。

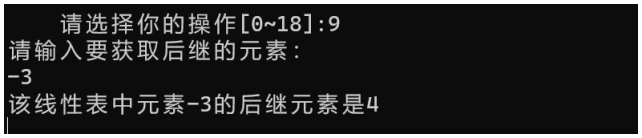

图 1-21　获取后继

测试用例：-2 1 -3 4 -1 2 1 -5 4，获取 6 的后继，结果图如下。

图 1-22　获取后继

10) 插入元素

测试用例：-2 1 -3 4 -1 2 1 -5 4，在第二位插入元素 9，结果如下图。

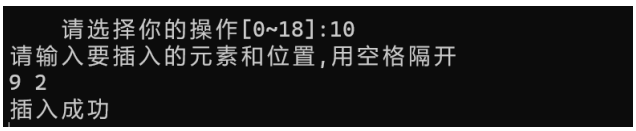

图 1-23　插入元素

之后遍历线性表，结果图如下。

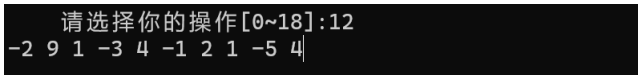

图 1-24　插入元素后遍历

11) 删除元素

测试用例：-2 9 1 -3 4 -1 2 1 -5 4，删除第二个元素，结果图如下。

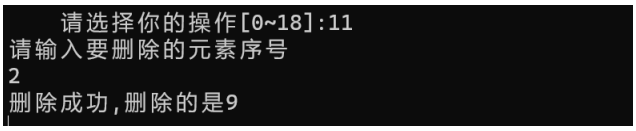

图 1-25　删除元素

之后遍历线性表，结果图如下。

图 1-26　删除元素后遍历

12) 遍历表

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

图 1-27　遍历表

13) 读取文件

测试用例：文件名 TEST，文件内容为-2 1 -3 4 -1 2 1 -5 4, 读取后遍历，结果图如下。

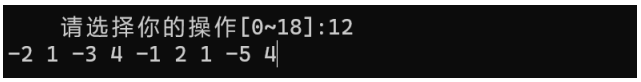

图 1-28　读取文件

14) 写入文件

测试用例：测试用例为-2 1 -3 4 -1 2 1 -5 4，结果图如下。

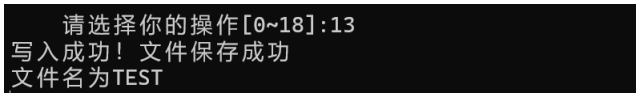

图 1-29　写入文件

15) 选择在第几个表上操作

测试用例：选择在第一个表上操作，结果图如下。

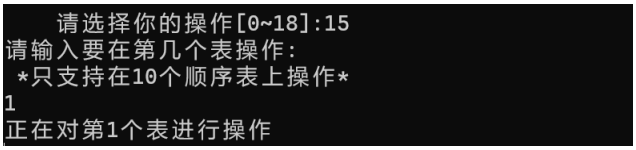

图 1-30　选择表

16) 连续子数组最大和

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

图 1-31　连续子数组最大和

17) 和为 k 的最大连续子数组个数

测试用例：-2 1 -3 4 -1 2 1 -5 4，求和为-1 的连续最大子数，结果图如下。

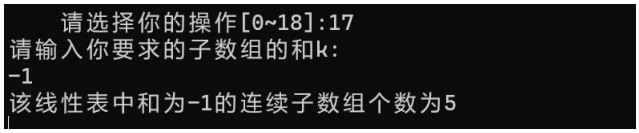

图 1-32　和为-1 的最大连续子数组个数

测试用例：-2 1 -3 4 -1 2 1 -5 4，求和为 20 的连续最大子数，结果图如下。

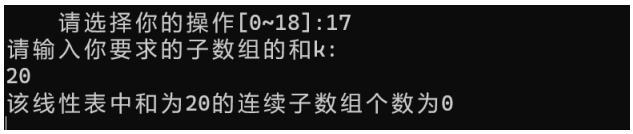

图 1-33　和为 20 的最大连续子数组个数

18) 从小到大排序

测试用例：-2 1 -3 4 -1 2 1 -5 4，结果图如下。

图 1-34　从小到大排序

排序后遍历表，结果图如下。

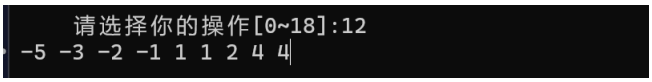

图 1-35　排序后遍历

1.5　实验小结

在本次实验中，我使用的语言为 C 语言，通过代码的设计和编写完成了线性表的顺序存储。在实验过程中，我理解了线性表的顺序物理存储结构的编写和实现原理，学会了线性表的初始化、遍历、查找等基本操作。

在调试中，我碰到的第一个错误是细节上拼写的错误。从老师给的模板上复制来的框架内容和预定义常量部分和实际的正确拼写有一点小小的偏差，比如 INFEASIBLE 拼成了 INFEASABLE，一字之差，就会导致程序无法运行，这提醒我了代码的严谨性是至关重要的。

实验过程中我遇到的最大的困难是关于文件的处理和多线性表操作。上个学期学习 C 语言时文件部分我掌握得不太牢固，导致编写与文件有关的函数时比较生疏。但是通过在网上查找资料和不断试错，我掌握了关于文件打开、关闭、读入等基本操作。关于多线性表操作，我一开始设计的是每次调用任何函数都要先输入一遍线性表的名字以定位到正确的线性表。但是我认为这样过于繁琐，于是在经过思考和请教同学之后改成了把多线性表的结构设定为类似数组的结构，先通过定位线性表的功能选择要操作的线性表的序号，之后的操作全部在这个选择的线性表上完成，这样就不用每个函数调用前都输入线性表名字了。

总体来说，通过这次实验，我对文件处理有了更深的了解，补上了以前的知识漏洞，也对多函数的模块化处理以及菜单显示有了直观的认识，还学会了多线性表操作的实现，很有收获。

2 附录 A 基于顺序存储结构线性表实现的源程序

```
\noindent
/* 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 INFEASTABLE -1
#define OVERFLOW -2

#define MAX_NUM 10
typedef int status;
typedef int ElemType; //数据元素类型定义

/*-------page 22 on textbook -------*/
#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;

    //线性表集合的定义Lists
```

```
/*-----page 19 on textbook ---------*/
status InitList(SqList& L);
status DestroyList(SqList& L);
status ClearList(SqList& L);
status ListEmpty(SqList L);
status ListLength(SqList L);
status GetElem(SqList L,int i,ElemType &e);
status LocateElem(SqList L,ElemType e);
status PriorElem(SqList L,ElemType cur,ElemType &pre);
status NextElem(SqList L,ElemType cur,ElemType &next);
status ListInsert(SqList& L,int i,ElemType e);
status ListDelete(SqList& L,int i,ElemType& e);
status ListTraverse(SqList L);
status SaveList(SqList &L,char FileName[]);
status LoadList(SqList &L,char FileName[]);
status MaxSubArray(SqList& L);
status SubArrayNum(SqList& L,ElemType k);
status SortList(SqList& L);
int main(void){
int i,pre,next;
int i_num=1;
SqList L[MAX_NUM+1];
    for(i=0;i<MAX_NUM+1;i++)
    {
        L[i].elem = NULL;
        L[i].listsize = 0;
        L[i].length = 0;
    }
int op=1;
    int n,e;
    char filename[30];
```

```
while(op){
system("cls"); printf("\n\n");
printf("      Menu for Linear Table On Sequence Structure \n");
printf("-------------------------------------------------\n");
printf("       1. InitList       11. ListDelete\n");
printf("       2. DertroyList    12. ListTraverse\n");
printf("       3. ClearList      13. SaveList\n");
printf("       4. ListEmpty      14. LoadList\n");
printf("       5. ListLength     15. LocateList\n");
printf("       6. GetElem        16. MaxSubArray\n");
printf("       7. LocateElem     17. SubArrayNum\n");
printf("          8. PriorElem      18. SortList\n");
printf("          9. NextElem       \n");
printf("         10. ListInsert     \n");
printf("-------------------------------------------------\n");
printf("    请选择你的操作[0~20]:");
scanf("%d",&op);
    switch(op){
   case 1:
     printf("请输入要创建的线性表名称: \n");
     scanf("%s",filename);
 if(InitList(L[i_num])==OK) {
  printf("线性表创建成功! \n");
  printf("请输入元素, 用空格隔开, 以0结束! \n");
        for(int i=0;i<L[i_num].listsize;i++){
           int num;
    scanf("%d",&num);
    if(num==0) break;
  else L[i_num].elem[L[i_num].length++]=num;
}
 }
```

```
 else printf("线性表创建失败! \n");
 getchar();getchar();
 break;
   case 2:
    if(L[i_num].elem==NULL)
         {
             printf("线性表不存在!\n");
             getchar();getchar();
             break;
         }
 if(DestroyList(L[i_num])==OK)
     printf("线性表销毁成功! \n");
     else printf("线性表销毁失败! \n");
 getchar();getchar();
 break;
   case 3:
 if(ClearList(L[i_num])==OK)
     printf("线性表清空成功! \n");
     else printf("线性表清空失败! \n");
 getchar();getchar();
 break;
   case 4:
     if(ListEmpty(L[i_num])==TRUE)
printf("线性表为空! \n");
 else if(ListEmpty(L[i_num])==INFEASIBLE)
printf("线性表不存在! \n");
 else printf("线性表不为空! \n");
 getchar();getchar();
 break;
   case 5:
     n=ListLength(L[i_num]);
```

```
printf("线性表长度为%d",n);
getchar();getchar();
break;
  case 6:
    printf("请输入你要获取的元素序号: \n");
    scanf("%d",&i);
    GetElem(L[i_num],i,e);
getchar();getchar();
break;
  case 7:
   printf("请输入你要查找的元素: \n");
   scanf("%d",&e);
   LocateElem(L[i_num],e);
getchar();getchar();
break;
  case 8:
    printf("请输入要获取前驱的元素: \n");
    scanf("%d",&e);
    PriorElem(L[i_num],e,pre);
getchar();getchar();
break;
  case 9:
    printf("请输入要获取后继的元素: \n");
    scanf("%d",&e);
    NextElem(L[i_num],e,next);
getchar();getchar();
break;
  case 10:
   printf("请输入要插入的元素和位置,用空格隔开\n");
    scanf("%d %d",&e,&i);
    if(ListInsert(L[i_num],i,e)==OK)
```

```
                    printf("插入成功\n");
 getchar();getchar();
 break;
   case 11:
     printf("请输入要删除的元素序号\n");
 scanf("%d",&i);
 ListDelete(L[i_num],i,e);
 getchar();getchar();
 break;
   case 12:
     ListTraverse(L[i_num]);
 getchar();getchar();
 break;
   case 13:
     if(SaveList(L[i_num], filename)==OK){
            printf("文件保存成功\n文件名为%s\n",filename);
         }
getchar();getchar();
         break;
   case 14:
     if(L[i_num].elem!=NULL){
            printf("线性表已存在! 无法加载! ");
            getchar();getchar();
            break;
         }
     printf("请输入要加载的文件名:\n ");
 scanf("%s", filename);
 if(LoadList(L[i_num], filename)==OK){
               printf("文件加载成功\n");
 }
else{
```

```
                printf("文件加载失败! ");
            }
            break;
        case 15:
    printf("请输入要在第几个表操作:\n ");
             printf("*只支持在%d个顺序表上操作*\n",MAX_NUM);
    scanf("%d",&i_num);
           if((i_num<1)||(i_num>MAX_NUM)){
printf("请选择正确范围! 默认对第一个线性表进行操作\n");
i_num=1;
   }
   printf("正在对第%d个表进行操作\n",i_num);
           getchar(); getchar();
         break;
   case 16:
     MaxSubArray(L[i_num]);
 getchar();getchar();
 break;
   case 17:
     printf("请输入你要求的子数组的和k:\n");
     scanf("%d",&i);
     SubArrayNum(L[i_num],i);
 getchar();getchar();
 break;
   case 18:
     SortList(L[i_num]);
 getchar();getchar();
 break;
case 0:
         break;
    return 0;
```

```
}//end of switch
  }//end of while
printf("欢迎下次再使用本系统! \n");
}//end of main()
/*--------page 23 on textbook --------------------*/
status InitList(SqList& L){
L.elem = (ElemType *)malloc( LIST_INIT_SIZE * sizeof (ElemType));
if(!L.elem) exit(OVERFLOW);
L.length=0;
L.listsize=LIST_INIT_SIZE;
return OK;
}

status DestroyList(SqList& L)
// 2.如果线性表L存在，销毁线性表L，释放数据元素的空间
{
if(L.elem) {//如果线性表L存在
        free(L.elem);  //释放数据元素空间
        L.elem = NULL; //返回未初始化状态
        L.length = 0;
        L.listsize = 0;
        return OK;
    }
    else return INFEASIBLE;
}

status ClearList(SqList& L)
// 3.如果线性表L存在，删除线性表L中的所有元素
{
     if(L.elem==NULL){
        return INFEASIBLE;
```

```
    }
    L.length=0;
    free(L.elem);
    return OK;
}

status ListEmpty(SqList L)
// 4.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==NULL){
        return INFEASIBLE;
    }
    if(L.length==0){
        return TRUE;
    }
    return FALSE;
}

status ListLength(SqList L)
// 5.如果线性表L存在，返回线性表L的长度
{
    if(L.elem) return L.length; //返回表长
    else return INFEASIBLE;
}

status GetElem(SqList L,int i,ElemType &e)
// 6.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
{
if(!L.elem){
```

```
printf("线性表不存在!\n");
return INFEASIBLE;
}
    if(i<1||i>L.length)
{
printf("输入的i值不规范!\n");
return ERROR;
}
    e=L.elem[i-1];
    printf("该线性表的第%d个元素是%d\n",i,e);
    return OK;
}

int LocateElem(SqList L,ElemType e)
// 7.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该序号
{
    if(L.elem){
        int i;
        //printf("%d\n",L.elem[0]);
        for(i=1;i<=L.length;i++){//遍历顺序表，查找指定元素e
            if (L.elem[i-1] == e) {
        printf("元素%d在该线性表中第一次出现的位置序号是%d\n",e,i);
        return i;//查找成功，则返回元素逻辑序号i
}
        }
        printf("元素%d在该线性表中不存在\n",e);
        return 0;//没有找到指定的元素e，查找失败，返回0
    }
    else {
printf("线性表不存在!\n");
return INFEASIBLE;
```

```
}
}

status PriorElem(SqList L,ElemType e,ElemType &pre)
// 8.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
{
    if (L.elem){
        int i;
        for(i=1;i<L.length;i++){//从第二个元素开始遍历线性表L
            if(L.elem[i] == e){ //若查找到e元素
                pre=L.elem[i-1];//获取线性表L中元素e的前驱
                printf("该线性表中元素%d的前驱元素是%d\n",e,pre);
                return OK;
            }
        }
        printf("无法找到元素%d的前驱\n",e);
        return ERROR;//没有前驱，返回ERROR
    }
    else {
printf("线性表不存在!\n");
return INFEASIBLE;
}
}

status NextElem(SqList L,ElemType e,ElemType &next)
// 9.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
{

if (L.elem){
        int i;
        for(i=0;i<L.length-1;i++){//遍历线性表L至倒数第二个元素
```

```
            if(L.elem[i] == e){ //若查找到e元素
                next=L.elem[i+1];//获取线性表L中元素e的后继
                printf("该线性表中元素%d的后继元素是%d\n",e,next);
                return OK;
            }
        }
        printf("无法找到元素%d的后继\n",e);
        return ERROR;//没有前驱，返回ERROR
    }
    else {
printf("线性表不存在!\n");
return INFEASIBLE;
}
}

status ListInsert(SqList &L,int i,ElemType e)
// 10.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
{
    //i的合法值为1<=i<=L.length+1
    if(i<1 || i>L.length+1) {                    //i值不合法
     printf("输入的i值不规范!\n");
return ERROR;
}
    if(L.elem){
        if(L.length>=L.listsize){   //当前存储空间已满，增加分配
        int *newbase = (ElemType *)realloc(L.elem,
        (L.listsize+LISTINCREMENT)*sizeof(ElemType));
            if(!newbase) exit(OVERFLOW);     //存储分配失败
            L.elem = newbase;                //新基址
            L.listsize += LISTINCREMENT;     //增加存储容量
        }
```

```
        int *q = &(L.elem[i-1]);    //q为插入位置
        for(int *p = &(L.elem[L.length-1]);p>=q;--p) *(p+1)=*p;
        //插入位置及之后的元素后移
        *q=e;           //插入e
        ++L.length;     //表长加一
        return OK;
    }
    else {
printf("线性表不存在!\n");
return INFEASIBLE;
}
}

status ListDelete(SqList &L,int i,ElemType &e)
// 11.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
{

if(L.elem){
if(i<1||i>L.length){
printf("删除位置不规范\n");
return ERROR;
}
        int *q = &(L.elem[i-1]);    //q为被删除元素的位置
        e=*q;                       //将第i个元素保存在e中
        ++q;
        for(int *p = &(L.elem[L.length-1]);p>=q;++q)
            *(q-1)=*q;
        //被删除元素位置及之后的元素前移
        --L.length;     //表长减一
        printf("删除成功,删除的是%d\n",e);
        return OK;
```

```
    }
    else {
printf("线性表不存在!\n");
return INFEASIBLE;
}
}

status ListTraverse(SqList L)
//12.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
{
if (L.elem){
        int i;
        if(L.length == 0){          //表长为0，为空线性表
            printf("空线性表");
            return 0;
        }
        for(i=0;i<L.length;i++){    //遍历线性表L
            if(i!=L.length-1)
                printf("%d ",L.elem[i]);
            else printf("%d",L.elem[i]);
        }
        return OK;
    }
    else {
printf("线性表不存在!\n");
return INFEASIBLE;
}
}


status  SaveList(SqList &L,char FileName[])
```

```
// 13.如果线性表L存在，将线性表L的的元素写到FileName文件中
{
    if(L.elem){
        FILE *fp;
        fp=fopen(FileName,"wb");      //以只读方式打开文件
        if(fp==NULL){                 //文件为空
            printf("文件为空!");
            exit (-1);
        }
        for(int i=0;i<L.length;i++){
            fprintf(fp,"%d " ,L.elem[i]);   //将线性表L的的元素写到文件中
        }
        fclose(fp);
        printf("写入成功! ");
        return OK;
    }
    else {
printf("线性表不存在!\n");
return INFEASIBLE;
}
}
status  LoadList(SqList &L,char FileName[])
// 14.如果线性表L不存在，将FileName文件中的数据读入到线性表L中
{
    FILE* p;
    int c,j=0;
    ElemType* newbase;

    if(L.elem){
        return INFEASIBLE;
    }
```

```
        L.elem=(ElemType*)malloc(sizeof(ElemType)*LIST_INIT_SIZE);
        L.length=0;
        L.listsize=LIST_INIT_SIZE;
        p=fopen(FileName ,"rb");
        if(p==NULL){
            printf("文件为空!");
            exit (-1);
        }
       while((fscanf(p,"%d",&c))!=EOF)   {
            if(L.length>=L.listsize){
                newbase=(ElemType*)realloc(L.elem,sizeof(ElemType)*
                (L.listsize+LISTINCREMENT));
                if(newbase==NULL){
                    return OVERFLOW;
                }
                L.elem=newbase;
                L.listsize+=LISTINCREMENT;
            }
            L.elem[j++]=c;
            L.length++;
        }
    fclose(p);
    return OK;
}


status MaxSubArray(SqList& L){
int sum,max=0;
if(!L.elem){
printf("线性表不存在!\n");
return INFEASIBLE;
```

```
}
for(int i=0;i<L.length;i++){
sum=0;
for(int j=0;j<L.length-i;j++){
sum+=L.elem[i+j];
if(sum>max) max=sum;
}
}
printf("该线性表的连续子数组的最大和为%d\n",max);
return max;
}

status SubArrayNum(SqList& L,ElemType k){
if(!L.elem){
printf("线性表不存在!\n");
return INFEASIBLE;
}
int sum,count=0;
for(int i=0;i<L.length;i++){
sum=0;
for(int j=0;j<L.length-i;j++){
sum+=L.elem[i+j];
if(sum==k) count++;
}
}
printf("该线性表中和为%d的连续子数组个数为%d\n",k,count);
return count;
}

status SortList(SqList& L){
if(!L.elem){
```

```
printf("线性表不存在!\n");
return INFEASIBLE;
}
int i,j,temp;
for(i=0;i<L.length-1;i++){
for(j=i+1;j<L.length;j++)
{
if(L.elem[i]>L.elem[j]){
temp=L.elem[i];
L.elem[i]=L.elem[j];
L.elem[j]=temp;
}
}
}
printf("排序成功! \n");
return OK;
}
```