

**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1703班**

**学 号： U201714661**

**姓 名： 吴迎楠**

**指导教师： 袁凌**

**报告日期： 2018年 11月 13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](#_Toc1363093817)

[1.1 问题描述](#_Toc2084263998)

[1.2 系统设计](#_Toc2144917531)

[1.2.1 系统总体设计](#_Toc1024438400)

[1.2.2 有关常量和类型定义](#_Toc1970817169)

[1.2.3 算法设计](#_Toc1730265926)

[1.3 顺序表演示系统实现与测试](#_Toc851316344)

[1.3.1 系统实现](#_Toc1883946786)

[1.3.2 系统测试](#_Toc607075315)

[1.4 实验小结](#_Toc1948272577)

**1 基于顺序存储结构的线性表实现**

**1.1 实验目的**

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 线性表基本运算定义

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，具体运算功能定义如下。

⑴初始化表：函数名称是InitaList(L)；初始条件是线性表L不存在已存在；操作结果是构造一个空的线性表。

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。

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。

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。

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。

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。

⑺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。

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。

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。

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。

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。

⑿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()。

**1.3系统设计**

1.3.1 系统总体设计

本系统采用顺序表作为线性表的物理结构，实现线性表的基本运算。

系统具有一个功能菜单。在主程序中完成函数调用所需参数值的准备和函数执行结果的现实，并给出适当的操作提示显示。

系统通过定义一个SqListsOptioner类型的含有线性表指针数组和当前线性表数量的结构体，并声明一个此类型的全局结构变量Lists，每当创建或者删除一个线性表，则修改此变量，实现对多个线性表的管理。

系统开始运行时调用函数OpenFiles()读取文件中的数据，并提供数据保存功能SaveData()以实现线性表的文件形式保存。

该演示系统提供的操作有：表的初始化、销毁、清空、判空，求表长、获取数据元素、查找数据元素、获得前驱、获得后继、插入数据元素、删除数据元素、表的遍历、表的选择、数据保存。

在程序中实现消息处理和操作提示，包括数据的输入和输出，错误操作提示、程序的退出。

1.3.2 有关常量和类型定义

#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 MAX\_SQLIST\_NUM 10 //可创建的线性表最大数量

typedef int status;

typedef int ElemType; //数据元素类型定义

typedef struct

{ //线性表（顺序结构）的定义

ElemType \* elem;

int length;

int listsize;

}SqList;

typedef struct

{ //多线性表信息结构定义（存储当前所有线性表地址及数量）

SqList \* SqListArray[MAX\_SQLIST\_NUM];

int length;

}SqListsOptioner;

SqLists Lists; //线性表信息表

1.3.3 算法设计

（1）InitList(SqList \* L)

设计：分配存储空间，并初始化表长为0，表容量为LIST\_INIT\_SIZE。每次创建表时新建的表位序为最大。假如当前SqListArrat数组中已经有3个表，那新建的表为第4个表。操作结果：构造一个空的线性表。

（2）DestroyList(SqList \* L)

设计：删除线性表内容并释放存储空间，每次操作当前线性表，销毁后当前线性表之后的线性表前移一个位序。例如当前操作表3，销毁表3后原表4左移成为表3，以此类推

操作结果：销毁线性表L。

（3）ClearList(SqList \* L)

设计：线性表L的长度赋值为0

操作结果：将L重置为空表。

（4）ListEmpty(SqList \*L)

设计：根据表长判断表是否为空

操作结果：若L为空表，则返回TRUE,否则返回FALSE。

（5）ListLength(SqList \*L)

设计：返回表长

操作结果：返回L中数据元素的个数。

（6）GetElem(SqList \*L, int i, ElemType \* e)

设计：根据位序找到第i个元素的地址并将其值赋值给指针e指向的元素

操作结果：用指针e指向的元素返回L中第i个数据元素的值。

（7）LocateElem(SqList \*L, ElemType e)

设计：遍历线性表找到第一个和元素e的相等的元素

操作结果：返回L中第1个与e相等的的数据元素的位序，若这样的数据元素不存在，则返回值为0。

（8）PriorElem(SqList \*L, ElemType cur, ElemType \* pre\_e)

设计：遍历线性表找到第一个和元素cur的相等的元素，如果其有前驱，用pre\_e返回，函数返回TRUE；否则函数返回FALSE，pre\_e无意义

操作结果：若cur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）NextElem（\*L，cur\_e，&next\_e）

设计：遍历线性表找到第一个和元素cur的相等的元素，如果其有后继，用next\_e返回，函数返回TRUE；否则函数返回FALSE，next\_e无意义

操作结果：若cur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）ListInsert(SqList \* L, int i, ElemType e)

设计：如果线性表已满，重新分配存储空间。将线性表指针L指向的线性表第i个元素之后的元素都右移一个位序，之后将e插入第i个位序

操作结果：在L的第i个位置之前插入新的数据元素e，L的长度加1

（11）ListDelete(SqList \* L, int i, ElemType \* e)

设计：将第i个位序的值赋给指针e指向的变量，之后第i个位序之后的元素全部左移一个位序

操作结果：删除L的第i个数据元素，用e返回其值，L的长度减1.

（12）ListTraverse(SqList L)

设计：在此处设置为遍历并打印表L中的每个元素值，返回表长

操作结果：依次输出表L中的每个变量的值

（13）PreLoadFiles(void)

设计：为了方便测试，在程序所在的目录里预先放好了两个文件”SqlistTest1.txt”和”SqlistTest2.txt”，在程序运行时会先读取这两个文件并将里面的线性表存放到内存中，成为可操作的线性表1和线性表2

操作结果：将预存文件内容写入到可操作的线性表中

（14）SaveData(void)

设计：首先函数将把线性表1 和线性表2这两个从预先存好的文件中获得的线性表存储回原文件，然后检查是否有用户新生成的线性表，如果有，则会提示用户并让用户手动输入文件名并创建新文件将线性表保存进去。

操作结果：保存程序中所有的线性表。

1. OpenFiles(void)

设计：当用户想要打开某文件时可以通过该函数手动输入文件名称来打开文件并自动将文件内的数据保存到新的线性表中。

操作结果：用户输入有效文件名，文件打开成功，数据被写入新线性表中

**1.4 顺序表演示系统实现与测试**

1.4.1 系统实现

运行环境：Windows下使用codeblocks编写程序并编译（默认MingGW编译器），程序清单如下：

>> File:SqList.h

#ifndef SQLIST\_H\_INCLUDED

#define SQLIST\_H\_INCLUDED

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define MAX\_SQLIST\_NUM 10//最多能同时操作的表的数目

typedef int status;

typedef int ElemType;

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

typedef struct//线性表结构

{

ElemType \*elem;

int length;

int listsize;

}SqList;

typedef struct//用来操作线性表的含线性表指针数组的结构

{

SqList\* SqListArray[MAX\_SQLIST\_NUM];

int length;

}SqListsOptioner;

SqListsOptioner Lists;//全局变量，用于操作

status InitiaList(SqList \*L);

status ListLength(SqList \*L);

status ListInsert(SqList \*L,int i,ElemType e);

status ListDelete(SqList \*L,int i,ElemType e);

status DestroyList(SqList \*L);

status CleaList(SqList \*L);

status ListEmpty(SqList \*L);

status GetElem(SqList \*L,int i,ElemType \*e);

status LocateElem(SqList \*L,ElemType e);

status PriorElem(SqList \*L,ElemType cur\_e,ElemType \*pre\_e);

status NextElem(SqList \*L,ElemType cur\_e,ElemType \*next\_e);

status ListTraverse(SqList \*L);//至此为实验要求中的12个函数

SqList \*ChooseList(int \*OpListNum);//多表操作中用来选择线性表

status PreLoadFiles(void);//打开预存的文件

status OpenFiles(void);//通过输入文件名的方式来打开文件

status SaveData(void);//保存文件

void menu(void);//打印菜单

#endif // SQLIST\_H\_INCLUDED

File: main.c

#include "SqList.h"

void main(void)

{

SqListsOptioner \*SqLists = &Lists;//头文件中的全局变量

SqList \*pList=NULL;//在程序中操作的线性表

int OpListNum=0;//当前操作的线性表的编号

int i;

int options;//在菜单中选择的选项

ElemType elem;//在后续操作中会用到的元素

if(!PreLoadFiles()) printf("\*\*\*\*程序运行错误！文件无法打开！\n");

printf("\*\*\*\*当前线性表数：%d\t（最大线性表数%d）\n",SqLists->length,MAX\_SQLIST\_NUM);

printf("\*\*\*\*当前操作的线性表（n=1,2,3....）:");//显示当前的线性表数量和正在操作的线性表

if(!OpListNum) printf("\t请选择要操作的线性表\n");

else printf("%d\n",OpListNum);

if(!SqLists->length) printf("\t（当前无可操作的线性表！）\n");

menu();//菜单

scanf("%d",&options);

getchar();

while(options)

{

system("cls");//每次循环后清屏

menu();

printf("\*\*\*\*当前线性表数：%d\t（最大线性表数%d）\n",SqLists->length,MAX\_SQLIST\_NUM);

printf("\*\*\*\*当前操作的线性表（n=1,2,3....）:\n");

switch(options)

{

case 0:

break;

case 1:

printf("\n\*\*\*\*你选择的是 InitiaList\n");

if(SqLists->length>=MAX\_SQLIST\_NUM)

{

printf("\t线性表操作数已达到最大值！不能继续创建线性表\n");

getchar();

break;

}

if (!(SqLists->SqListArray[SqLists->length++] =(SqList \*)malloc(sizeof(SqList))))

exit(OVERFLOW);

if(InitiaList(SqLists->SqListArray[SqLists->length-1])==OK)//在线性表指针数组中创建新指针并对该指针创建线性表

{

printf("\t线性表%d创建成功！\n",SqLists->length);

printf("\*\*\*\*当前线性表数：%d\t（最大线性表数%d）\n",SqLists->length,MAX\_SQLIST\_NUM);

}

else

{

printf("\t线性表创建失败！\n");

free(SqLists->SqListArray[SqLists->length-1]);

}

break;

case 2:

printf("\n\*\*\*\*你选择的是 DestroyList\n");

if(OpListNum<1)

{

printf("\t请先选择要销毁的线性表！\n");

break;

}

else if(DestroyList(pList))

{

printf("\t线性表%d销毁成功！\n",OpListNum);

if(OpListNum<SqLists->length)

for(i=OpListNum; i<SqLists->length-1; i++)

SqLists->SqListArray[i-1]=SqLists->SqListArray[i];

SqLists->length--;

}

OpListNum=0;

break;

case 3:

printf("\n\*\*\*\*你选择的是 ClearList\n");

if(OpListNum<1)

{

printf("\t请先选择要置空的线性表！\n");

break;

}

if(CleaList(pList))

printf("\t线性表%d置空成功！\n");

break;

case 4:

printf("\n\*\*\*\*你选择的是 ListEmpty\n");

if(OpListNum<1)

{

printf("\t请先选择要判空的线性表！\n");

break;

}

if(ListEmpty(pList))

printf("\t线性表为空！\n");

else printf("\t线性表不为空！\n");

break;

case 5:

printf("\n\*\*\*\*你选择的是 ListLength \n");

if(OpListNum<1)

{

printf("\t请先选择要求长度的线性表！\n");

break;

}

printf("\t线性表的长度为%d!\n",ListLength(pList));

break;

case 6:

printf("\n\*\*\*\*你选择的是 GetElem\n");

if(OpListNum<1)

{

printf("\t请先选择要获取元素的线性表！\n");

break;

}

printf("\t请输入你需要获得第几个元素：");

scanf("%d",&i);

getchar();

if(i>=1&&i<=pList->length)

{

GetElem(pList,i,&elem);

printf("\t线性表中的第%d个元素为：%d\n",i,elem);

}

else printf("\t元素不存在！\n");

break;

case 7:

printf("\n\*\*\*\*你选择的是 LocatElem\n");

if(OpListNum<1)

{

printf("\t请先选择要定位元素的线性表！\n");

break;

}

printf("\t请输入要查找的元素：");

scanf("%d",&elem);

getchar();

if(i=LocateElem(pList,elem)) printf("\t你查找的元素为表中的第%d个元素\n",i);

else printf("\t元素不存在！\n");

break;

case 8:

printf("\n\*\*\*\*你选择的是 PriorElem\n");

if(OpListNum<1)

{

printf("\t请先选择要获取元素前驱的线性表！\n");

break;

}

printf("\t请输入你要获取哪一个元素的前驱：");

scanf("%d",&i);

getchar();

if(PriorElem(pList,i,&elem)) printf("\t元素%d的前驱是%d！\n",i,elem);

else printf("\t前驱不存在！\n");

break;

case 9:

printf("\n\*\*\*\*你选择的是 NextElem\n");

if(OpListNum<1)

{

printf("\t请先选择要获取元素后继的线性表！\n");

break;

}

printf("\t请输入你要获取哪一个元素的后继：");

scanf("%d",&i);

getchar();

if(NextElem(pList,i,&elem)) printf("\t元素%d的后继是%d！\n",i,elem);

else printf("\t后继不存在！\n");

break;

case 10:

printf("\n\*\*\*\*你选择的是 ListInsert\n");

if(OpListNum<1)

{

printf("\t请先选择要插入元素的线性表！\n");

break;

}

printf("\t请输入你要插入元素的位置和要插入元素的值：");

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

getchar();

if(ListInsert(pList,i,elem)) printf("\t元素%d已成功插入到第%d个位置之前！\n",elem,i);

else printf("\t插入失败！\n");

break;

case 11:

printf("\n\*\*\*\*你选择的是 ListDelete\n");

if(OpListNum<1)

{

printf("\t请先选择要删除元素的线性表！\n");

break;

}

printf("\t请输入你要删除的元素位置：");

scanf("%d",&i);

getchar();

if(ListDelete(pList,i,&elem)) printf("\t第%d个元素%d已从线性表中删除！\n",i,elem);

else printf("\t元素删除失败！\n");

break;

case 12:

printf("\n\*\*\*\*你选择的是ListTrabverse\n");

if(OpListNum<1)

{

printf("\t请先选择要遍历的线性表！\n");

break;

}

if(!ListTraverse(pList)) printf("\t线性表为空表！\n");

break;

case 13:

printf("\n\*\*\*\*你选择的是ChooseList\n");

pList=ChooseList(&OpListNum);

printf("\t当前操作的线性表为线性表%d\n",OpListNum);

break;

case 14:

printf("\n\*\*\*\*你选择的是SaveData\n");

if(SaveData()) printf("\t文件保存成功！\n");

else printf("\t文件保存失败！\n");

break;

case 15:

printf("\n\*\*\*\*你选择的是OpenFiles\n");

if(OpenFiles()) printf("\t文件打开成功！\n");

else printf("\t文件打开失败！\n");

break;

default:

printf("\n\t指令错误！\n");

}

scanf("%d",&options);//读入下一个指令

getchar();

}

printf("\t欢迎下次再使用本系统！\n");

}

>> File: implements.c

#include "SqList.h"

status InitiaList(SqList \*L)//构造新的线性表

{

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 ListLength(SqList \*L)//求表长,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。

{

return L->length;

}

status ListInsert(SqList \*L,int i,ElemType e)//在顺序线性表L中第i个位置之前插入新元素e

{

//i的合法值为 1<=i<=ListLength(\*L)+1

if(i<1||i>L->length+1) return ERROR;

ElemType \*q,\*p;

if(L->length>=L->listsize)

{

ElemType \*newbase;

newbase = (ElemType \*)realloc(L->elem,(L->listsize+LISTINCREMENT)\*sizeof(ElemType));

if(!newbase) exit(OVERFLOW);

L->elem=newbase;

L->listsize+=LISTINCREMENT;

}

q=&(L->elem[i-1]);//此时q为插入位置

for(p=&(L->elem[L->length-1]); p>=q; --p)

\*(p+1)=\*p;

\*q=e;

++L->length;

return OK;

}

status ListDelete(SqList \*L,int i,ElemType \*e)//删除元素,初始条件是线性表L已存在且非空,删除L的第i个数据元素，用e返回其值。

{

//i的合法值为1≤i≤ListLength(L)

if(i<1||(i>L->length)) return ERROR;

ElemType \*p,\*q;

p=&(L->elem[i-1]);

\*e=\*p;

q=L->elem+L->length-1;

for(++p; p<=q; ++p)

\*(p-1)=\*p;

--L->length;

return OK;

}

status DestroyList(SqList \*L)//销毁表：初始条件是线性表L已存在；操作结果是销毁线性表L。

{

free(L->elem);

free(L);

return OK;

}

status CleaList(SqList \*L)//清空表：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。

{

L->length=0;

return TRUE;

}

status ListEmpty(SqList \*L)//判定空表：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。

{

if(L->length==0) return TRUE;

else return FALSE;

}

status GetElem(SqList \*L,int i,ElemType \*e)//获得元素：初始条件是线性表已存在，操作结果是用e返回L中第i个数据元素的值。

{

//i的合法值范围是1≤i≤ListLength(L)

\*e = L->elem[i-1];

return OK;

}

status LocateElem(SqList \*L,ElemType e)//查找元素：返回第一个值为e的位序，没有则返回0

{

int i;

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

if(e == L->elem[i]) return i+1;

return 0;

}

status PriorElem(SqList \*L,ElemType cur\_e,ElemType \*pre\_e)//获得前驱，并用pre\_e返回

{

//cur\_e是L的数据元素，且不是第一个,否则操作失败，pre\_e无定义

int i;

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

if(cur\_e==L->elem[i])

{

\*pre\_e = L->elem[i-1];

return OK;

}

return FALSE;

}

status NextElem(SqList \*L,ElemType cur\_e,ElemType \*next\_e)//获得后继，并用next\_e返回

{

//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。

int i;

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

if(cur\_e==L->elem[i])

{

\*next\_e=L->elem[i+1];

return OK;

}

return FALSE;

}

status ListTraverse(SqList \*L)//遍历表：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()。

{

int i;

printf("\n---------------------所有元素 -----------------------\n");

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

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

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

return L->length;

}

SqList\* ChooseList(int \*OpListNum)//选择要操作的线性表

{

printf("\t请输入要选择的线性表编号(1-%d)：",Lists.length);

int choice;

scanf("%d",&choice);

getchar();

if(choice<=0||choice>Lists.length)

{

printf("该线性表不存在！\n");

return NULL;

}

else

{

\*OpListNum=choice;

return Lists.SqListArray[\*OpListNum-1];

}

}

status PreLoadingFiles(void)//打开预存的两个文件并将该文件内容读取到线性表中

{

FILE \*pFile1,\*pFile2;

SqListsOptioner \*SqLists=&Lists;

SqList \*pSqList;

if((pFile1=fopen("SqlistTest1.txt","r+"))==NULL)//读取文件1

{

printf("\t文件读取失败！\n");

return 0;

}

else

{

int i=1,p;

printf("\t文件1打开成功！");

SqLists->SqListArray[SqLists->length] = (SqList \*)malloc(sizeof(SqList\*));

pSqList=SqLists->SqListArray[SqLists->length];

InitiaList(pSqList);

SqLists->length++;

while(fscanf(pFile1,"%d",&p)!=EOF)

{

if(ListInsert(pSqList,i,p)==OK)i++;

else printf("读取失败！");

}

printf("信息读取完成！\n");

}

if((pFile2=fopen("SqlistTest2.txt","r+"))==NULL)//读取文件2

{

printf("\t文件读取失败！\n");

return 0;

}

else

{

int i=1,p;

printf("\t文件2打开成功！");

SqLists->SqListArray[SqLists->length] = (SqList \*)malloc(sizeof(SqList\*));

pSqList=SqLists->SqListArray[SqLists->length];

InitiaList(pSqList);

SqLists->length++;

while(fscanf(pFile2,"%d",&p)!=EOF)

{

if(ListInsert(pSqList,i,p)==OK)i++;

else printf("读取失败！");

}

printf("信息读取完成！\n");

fclose(pFile2);

}

return OK;

}

status OpenFiles(void)//允许用户通过输入文件名来打开文件

{

SqListsOptioner \*SqLists=&Lists;

SqList \*pSqList;

FILE \*pFile;

char FileName[40];

int i=1,p;

printf("\t请输入要打开文件的文件名:\n\t");

scanf("%s",FileName);

getchar();

if((pFile=fopen(FileName,"r+"))==NULL)

{

printf("\*\*\*\*文件打开失败！\n");

return FALSE;

}

else

{

SqLists->SqListArray[SqLists->length] = (SqList \*)malloc(sizeof(SqList\*));

pSqList=SqLists->SqListArray[SqLists->length];

InitiaList(pSqList);

SqLists->length++;

while(fscanf(pFile,"%d",&p)!=EOF)

{

if(ListInsert(pSqList,i,p)==OK)i++;

else

{

printf("\*\*\*\*读取失败！");

return FALSE;

}

}

printf("\t信息读取完成！\n");

printf("\t当前线性表数%d\n",SqLists->length);

}

return OK;

}

status SaveData(void)

{

FILE \*pFILE;

SqListsOptioner \*SqLists = &Lists;

SqList \*pList;

int i,rest,Saving\_num=3;

if((pFILE = fopen("SqlistTest1.txt", "w+"))==NULL) return FALSE;//将内存中的线性表数据1,2写入到文件1，2中

pList=SqLists->SqListArray[0];

for(i=0; i<pList->length; i++)

fprintf(pFILE," %d",pList->elem[i]);

fclose(pFILE) ;

if((pFILE= fopen("SqlistTest2.txt", "w+"))==NULL) return FALSE;

pList=SqLists->SqListArray[1];

for(i=0; i<pList->length; i++)

fprintf(pFILE," %d",pList->elem[i]);

fclose(pFILE) ;

for(rest=(SqLists->length)-2; rest>0; rest--)//检查是否有新建的线性表

{

char FileName[40];

printf("\t还有%d个新建的线性表待保存！请输入线性表%d的文件名：\n",rest,Saving\_num);

scanf("%s",FileName);

getchar();

if((pFILE=fopen(FileName,"w+"))==NULL) return FALSE;

pList=SqLists->SqListArray[Saving\_num-1];

for(i=0; i<pList->length; i++)

fprintf(pFILE," %d",pList->elem[i]);

fclose(pFILE);

Saving\_num++;

}

return OK;

}

void menu(void) //打印菜单

{

printf("\n\n");

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

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

printf(" 1. InitiaList 7. LocatElem\n");

printf(" 2. DestroyList 8. PriorElem\n");

printf(" 3. ClearList 9. NextElem \n");

printf(" 4. ListEmpty 10. ListInsert\n");

printf(" 5. ListLength 11. ListDelete\n");

printf(" 6. GetElem 12. ListTrabverse\n");

printf(" 0. Exit 13.ChooseList \n");

printf(" 14.SaveData 15.OpenFiles \n");

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

printf("\n\t\t请输入你的选择:\n");

}

1.4.2 系统测试

测试数据

表1：

1 2 3 4 5

表2

5 4 3 2 1

表3

25

测试用例及其结果如下（各函数测试为独立测试，测试初始数据相同，不受上个函数测试影响）：

1. 测试函数：ChooseList

测试步骤及结果如表1-1所示

表1-1 ChooseList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |

1. 测试函数：InitiaList

测试步骤及结果如表1-2所示

表1-2 InitiaList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入1进入函数InitiaList | 因为预加载了两个线性表，所以新线性表为线性表3。按回车后输出“线性表3创建成功！” | 按回车后输出“线性表3创建成功！” |
| 2 | 1. 主界面输入13进入函数ChooseList 2. 按提示输入要操作的线性表序号，输入3 | 按回车后输出“当前线性表为线性表3”当前操作线性表更新为3 | 按回车后输出“当前线性表为线性表3”当前操作线性表更新为3 |
| 3 | 主界面输入12进入函数ListTrabverse | 按回车后输出“  --所有元素--  --结束--  线性表为空表！ | 按回车后输出“  --所有元素--  --结束--  线性表为空表！ |
| 4 | 1.主界面输入4进入函数ListEmpty | 按回车后输出“线性表为空！” | 按回车后输出“线性表为空！” |

1. 测试函数：DestroyList

测试步骤及结果如表1-3所示

表1-3 DestroyList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为1 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  1 2 3 4 5  --结束--” | 输出“---所有元素--  1 2 3 4 5  --结束--” |
| 3 | 主界面输入2进入函数DestroyList | 按回车后输出“线性表销毁成功！”当前线性表数更新为1，当前操作线性表更新为无，原线性表2成为新线性表1 | 按回车后输出“线性表销毁成功！”当前线性表数更新为1，当前操作线性表更新为无，原线性表2成为新线性表1 |
| 4 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为1 |
| 5 | 主界面输入12进入函数ListTrabverse | 原表2变为表1，输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |

1. 测试函数：ClearList

测试步骤及结果如表1-4所示

表1-4 ClearList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为1 |
| 2 | 主界面输入4进入函数ListEmpty | 输出“线性表不为空！” | 输出“线性表不为空！” |
| 3 | 主界面输入3进入函数ClearList | 输出“线性表1置空成功！” | 输出“线性表1置空成功！” |
| 4 | 主界面输入4进入函数ListEmpty | 输出“线性表为空！” | 输出“线性表为空！” |

1. 测试函数：ListEmpty

测试步骤及结果如表1-5所示

表1-5 ListEmpty函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 主界面输入4进入函数ListEmpty | 输出“线性表不为空！” | 输出“线性表不为空！” |
| 4 | 主界面输入3进入函数ClearList | 输出“线性表2置空成功！” | 输出“线性表2置空成功！” |
| 5 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--    ---结束---  线性表为空表！” | 输出“---所有元素--    ---结束---  线性表为空表！” |
| 6 | 主界面输入4进入函数ListEmpty | 输出“线性表为空！” | 输出“线性表为空！” |

1. 测试函数：ListLength

测试步骤及结果如表1-6所示

表1-6 ListLength函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 主界面输入5进入函数ListLength | 输出“线性表的长度为5!” | 输出“线性表的长度为5!” |

1. 测试函数：GetElem

测试步骤及结果如表1-7所示

表1-7 GetElem函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入6进入函数GetElem  2.按提示输入要查找的元素位序，输入4 | 输出“线性表中的第4个元素为：2” | 输出“线性表中的第4个元素为：2” |

1. 测试函数：PriorElem

测试步骤及结果如表1-8所示

表1-8 PriorElem函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入8进入函数PriorElem  2.按提示输入要查找的元素值，输入4 | 输出“元素4的前驱是5！” | 输出“元素4的前驱是5！” |
| 4 | 1.主界面输入8进入函数PriorElem  2.按提示输入要查找的元素值，输入5 | 输出“前驱不存在!” | 输出“前驱不存在!” |

1. 测试函数：NextElem

测试步骤及结果如表1-9所示

表1-9 NextElem函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入9进入函数NextElem  2.按提示输入要查找的元素值，输入4 | 输出“元素4的后继是3！” | 输出“元素4的后继是3！” |
| 4 | 1.主界面输入9进入函数NextElem  2.按提示输入要查找的元素值，输入1 | 输出“后继不存在!” | 输出“后继不存在!” |

1. 测试函数：ListInsert

测试步骤及结果如表1-10所示

表1-10 ListInsert函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入10进入函数ListInsert  2.按提示输入要插入元素的位置，输入1  3. 按提示输入要插入的元素值，输入25 | 输出“元素25已成功插入第1个位置之前！” | 输出“元素25已成功插入第1个位置之前！” |
| 4 | 主界面输入12进入函数ListTrabverse | 输出“25 5 4 3 2 1” | 输出“25 5 4 3 2 1” |
| 5 | 1.主界面输入10进入函数ListInsert  2.按提示输入要插入元素的位序，输入10（超出线性表长度）  3. 按提示输入要插入的元素值，输入7 | 输出“插入失败！” | 输出“插入失败！” |
| 6 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  25 5 4 3 2 1  --结束--” | 输出“---所有元素--  25 5 4 3 2 1  --结束--” |

1. 测试函数：ListDelete

测试步骤及结果如表1-11所示

表1-11 ListDelete函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1. 主界面输入13进入函数ChooseList   2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入11进入函数ListDelete  2.按提示输入要删除的位序，输入2 | 输出“第2个元素4已从线性表中删除！” | 输出“第2个元素4已从线性表中删除！” |
| 4 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 3 2 1  --结束--” | 输出“---所有元素--  5 3 2 1  --结束--” |
| 5 | 1.主界面输入11进入函数ListDelete  2.按提示输入要删除的位序，输入1 | 输出“第1个元素5已从线性表中删除！” | 输出“第1个元素5已从线性表中删除！” |
| 6 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  3 2 1  --结束--” | 输出“---所有元素--  3 2 1  --结束--” |
| 7 | 1.主界面输入11进入函数ListDelete  2.按提示输入要删除的位序，输入100 | 输出“删除失败！” | 输出“删除失败！” |

1. 测试函数：ListTraverse

测试步骤及结果如表1-12所示

表1-12 ListTraverse函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseLIst  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为表1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为表1 |
| 4 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  1 2 3 4 5  --结束--” | 输出“---所有元素--  1 2 3 4 5  --结束--” |

1. 测试函数：ChooseList

测试步骤及结果如表1-13所示

表1-13 ChooseList函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1.主界面输入13进入函数ChooseLIst  2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入13进入函数ChooseList  2.按提示输入要操作的线性表序号，输入1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为表1 | 按回车后输出“当前线性表为线性表1”当前操作线性表更新为表1 |
| 4 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  1 2 3 4 5  --结束--” | 输出“---所有元素--  1 2 3 4 5  --结束--” |

1. 测试函数：SaveData

测试步骤及结果如表1-14所示

表1-14SaveData函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1. 主界面输入13进入函数ChooseList   2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 2 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 4 3 2 1  --结束--” | 输出“---所有元素--  5 4 3 2 1  --结束--” |
| 3 | 1.主界面输入11进入函数ListDelete  2.按提示输入要删除的位序，输入2 | 输出“第2个元素4已从线性表中删除！” | 输出“第2个元素4已从线性表中删除！” |
| 4 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 3 2 1  --结束--” | 输出“---所有元素--  5 3 2 1  --结束--” |
| 5 | 主界面输入14进入函数SaveData | 输出“文件保存成功！” | 输出“文件保存成功！” |
| 6 | 重新运行目标程序 |  |  |
| 7 | 1. 主界面输入13进入函数ChooseList   2.按提示输入要操作的线性表序号，输入2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 | 按回车后输出“当前线性表为线性表2”当前操作线性表更新为表2 |
| 8 | 主界面输入12进入函数ListTrabverse | 输出“---所有元素--  5 3 2 1  --结束--” | 输出“---所有元素--  5 3 2 1  --结束--” |
| 9 | 主界面输入1进入函数InitiaList | 因为预加载了两个线性表，所以新线性表为线性表3。按回车后输出“线性表3创建成功！” | 按回车后输出“线性表3创建成功！” |
| 10 | 1. 主界面输入13进入函数ChooseList 2. 按提示输入要操作的线性表序号，输入3 | 按回车后输出“当前线性表为线性表3”当前操作线性表更新为表3 | 按回车后输出“当前线性表为线性表3”当前操作线性表更新为表3 |
| 11 | 1.主界面输入10进入函数ListInsert  2.按提示输入要插入元素的位置，输入1  3. 按提示输入要插入的元素值，输入25 | 输出“元素25已成功插入第1个位置之前！” | 输出“元素25已成功插入第1个位置之前！” |
| 12 | 输入14进入函数SaveData | 输出“还有一个新建的线性表待保存！请输入线性表3的文件名：” | 输出“还有一个新建的线性表待保存！请输入线性表3的文件名：” |
| 13 | 按提示输入文件名”Test.txt”并按回车 | 输出“文件保存成功！” | 输出“文件保存成功！” |
| 14 | 打开程序所在目录 | 发现新增文件Test.txt，打开后内容为25 | 发现新增文件Test.txt，打开后内容为25 |

1. 测试函数：OpenFiles

测试步骤及结果如表1-15所示

表1-15 OpenFiles函数测试

|  |  |  |  |
| --- | --- | --- | --- |
| 测试步骤 | 测试输入 | 理论结果 | 运行结果 |
| 1 | 1. 主界面输入15进入函数OpenFiles 2. 按照提示输入要打开的文件名，输入Test.txt（其内数据为25） | 输出“信息读取完成！当前线性表数3  文件打开成功” | 输出“信息读取完成！当前线性表数3  文件打开成功” |
| 2 | 1. 主界面输入13进入函数ChooseList 2. 按提示输入要操作的线性表序号，输入3 | 按回车后输出“当前线性表为线性表3”当前操作线性表更新为表3 | 按回车后输出“当前线性表为线性表3”当前操作线性表更新为表3 |
| 3 | 1.主界面输入12进入函数ListTrabverse | 输出“---所有元素--  25  --结束--” | 输出“---所有元素--  25  --结束--” |

**1.4 实验小结**

本次实验加深了我对线性表的概念、基本运算的理解，掌握了线性表的基本预算的实现，熟练了线性表的逻辑结构和物理结构之间的关系。同时，我也学习到了程序设计过程中应该考虑的用户交互和用户体验，并从多方面考虑用户输入错误的情况下应该如何处理和应对。

今后的学习过程中应当多从数据结构的角度分析如何进行数据的处理、存储以方便问题的解决，并要勤加练习达到熟能生巧的地步。