

**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201707**

**学 号： U201714786**

**姓 名： 王占成**

**指导教师： 祝建华**

**报告日期：**

**计算机科学与技术学院**

# 1 基于顺序存储结构的线性表实现

## 问题描述

### 实验目标：

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。

### 实验内容：

采用顺序表作为线性表的物理结构，实现多种对表的操作。同时本程序具有菜单演示的功能。其中，在主程序中完成函数调用所需实参值的准备和函数执行结果的显示，并且给出适当的操作提示。演示系统实现了多个线性表管理，并且在不同的表中进行切换，来真正地实现对表的多种操作。

线性表在物理内存中可以以顺序表的方式实现，即物理上存储位置相邻的两个元素是线性表中的相邻元素，且数据元素的前后关系不变。

实验要完成的顺序表算法：

1. 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。

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,a)；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有相等学号的学生的位序。

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已存在且非空，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)，初始条件是线性表L已存在；操作结果是输出表中每个数据元素的值。

13． 读取文件：函数名称是ReadFromFile(\*L,moro,\*Isinit,\*hw,wh)，初始条件是线性表L已存在，操作结果是将txt文件中的内容读取到顺序表中。

14. 创建多表：没有具体函数，无初始条件，操作结果是创建一个顺序表来储存多个顺序表。

15. 切换多表：无具体函数，初始条件是多表已被创建，操作结果是将当前操作的表切换为其它表。

16. 保存文件：函数名称是WriteToFile(\*L)，初始条件是线性表L已存在，操作结果是将表中的内容保存在txt文件中。

## 系统设计

演示系统可进行单表和多表的操作，在单表操作的过程中可切换为多表操作。多表之间可以进行切换，切换过程中可选择保存当前表或者在最后退出程序的时候进行保存。当选择对当前的表进行保存时，可自由的命名文件名，文件将最终以txt文件的形式保存在文件夹中。（如图1-1所示）

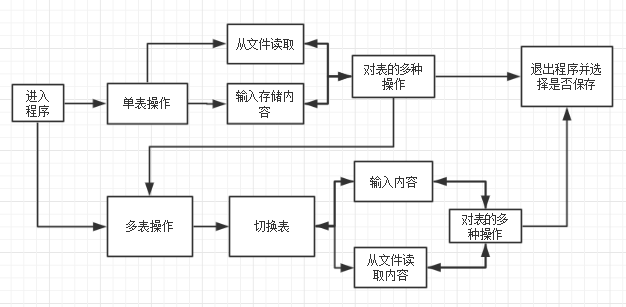


图1-1 程序的系统设计流程图

## 系统实现

在进入系统时，首先默认的操作为单表操作，此时表未初始化。此时可以进行三种操作：

### 1.3.1 初始化表

初始化表，并且向表中键入内容，然后可以进行对表的任何操作。其中比较特别的时进入多表操作和从文件读取内容。

#### 1.3.1.1 多表操作

此时需要选择是否保存单表，若单表不保存，默认为被舍弃。若单表被保存，则需要输入文件名。之后进入多表操作，选择要同时操作表的数目，之后可进行与单表同样的操作(注意：当多表已经创建时，不可以再次创建多表)。当在不同的表进行切换时，选择是否保存当前表，若不保存，可以在程序退出时再次选择是否保存。退出当前表时，程序将询问是否保存当前表，若保存，则输入文件名；否则，舍弃当前表。切换表时不可再次切换到该表。之后程序将自动切换到其他的表，当所有的表被保存或被舍弃时，则退出程序。

#### 1.3.1.2 从文件读取内容

此时需要输入文件名，若文件存在，则自动的清空表并且将文件的内容读取到表中，之后可以进行与单表操作同样的操作。

若文件不存在，则读取失败。若想重新向表中输入内容，需要销毁表并且重新初始化。

### 1.3.2 读取文件

系统自动初始化表，并且从文件中读取内容。此时可以进行与单表操作同样的操作。

### 1.3.3 进行多表操作

直接进入多表操作。因为并没有初始化单表，所以并不会询问是否保存单表。然后可以进行与1.1 同样的操作。

### 程序源代码如下：

/\*\*

\* 本次代码实现的原理如下：

\* 开辟一段连续的内存空间来储存指针

\* 每个指针指向不同的学生

\* 每个学生信息的储存均为一段连续的内存空间

\*/

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <iostream>

#include <cstring>

#include <fstream>

using namespace std;

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define OVERFLOW -2

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define EVERY 20

typedef int status; //数据元素类型定义

typedef struct {

char id[EVERY];

char name[EVERY];

float score;

} ElemType;

typedef struct { //顺序表的定义

ElemType \*elem = nullptr;

int length;

int listsize;

} SqList;

status IntiaList(SqList \*L);

status DestroyList(SqList \*L);

status ClearList(SqList \*L);

status ListEmpty(SqList L);

int ListLength(SqList L);

status GetElem(SqList L, char \*stu, ElemType \*e);

status LocateElem(SqList L, ElemType e,int &a);

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

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

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

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

status ListTrabverse(SqList L);

status ReadFromFile(SqList \*L, int moro, int \*Isinit, int \*hw, int wh);

status WriteToFile(SqList \*L);

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

int main(void) {

SqList Q;

SqList \* L = &Q;

int choice = 1;

int Isinit = 0;

int hm = 0;//用来保存多表的数目

int wh = 0;//定位当前表在多表中哪一个表

int moro = 0;//判断是多表操作还是单表操作

SqList \* multilist = nullptr;

int \*hw = nullptr;//保存表的状态 0：未初始化 1：初始化 2：已保存 3：不保存

while (choice) {

cout << "\n\n" << endl;

cout <<" Menu for Linear Table On Sequence Structure \n" << endl;

cout <<"-------------------------------------------------\n" << endl;

cout <<" 1. IntiaList 9. NextElem\n" << endl;

cout <<" 2. DestroyList 10. ListInsert\n" << endl;

cout <<" 3. ClearList 11. ListDelete\n" << endl;

cout <<" 4. ListEmpty 12. ListTrabverse\n" << endl;

cout <<" 5. ListLength 13. ReadFromFile\n" << endl;

cout <<" 6. GetElem 14. MultiList\n" << endl;

cout <<" 7. LocateElem 15. SwitchList\n" << endl;

cout <<" 8. PriorElem 16. SaveList\n" << endl;

cout <<" 0. Exit\n" << endl;

cout <<"-------------------------------------------------\n" << endl;

if(moro == 1)

cout << "当前操作的为表 " << wh << endl;

cout <<" 请选择你的操作[0~15]:" << endl;

cin >> choice;

switch (choice) {

case 1: {

if(moro == 1){

Isinit = hw[wh - 1];

}

if(Isinit == 1){

cout << "当前已初始化表!" << endl;

break;

}

if (IntiaList(L)) {

if(moro == 1) {

hw[wh - 1] = 1;

Isinit = hw[wh - 1];

}

else

Isinit = 1;

int number;

cout << "线性表创建成功！" << endl;

cout << "输入学生的信息，依次输入学号、姓名、得分(学生的个数不大于" << LIST\_INIT\_SIZE << ")，学号中输入符号#结束输入" << endl;

for (number = 0; number <= LIST\_INIT\_SIZE; number++) {

cout << "请输入第" << number + 1 << "个学生的信息" << endl;

cin.sync();

cin >> (\*L).elem[number].id;

if (number == LIST\_INIT\_SIZE && strcmp((\*L).elem[number].id,"#") != 0) {

cout << "超出范围!" << endl;

break;

}

if (strcmp((\*L).elem[number].id,"#") == 0)

break;

cin >> (\*L).elem[number].name;

cin >> (\*L).elem[number].score;

(\*L).length++;

}

} else

cout << "线性表创建失败！" << endl;

system("pause");

break;

}

case 2: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

break;

}

if (DestroyList(L)) {

cout << "销毁顺序表成功" << endl;

Isinit = 0;

}

system("pause");

break;

}

case 3: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

if (ClearList(L))

cout << "清空顺序表成功" << endl;

system("pause");

break;

}

case 4: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

if (ListEmpty(\*L))

cout << "顺序表为空" << endl;

else {

cout << "顺序表不为空" << endl;

}

system("pause");

break;

}

case 5: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

cout << "顺序表的长度为 " << ListLength(\*L) << endl;

system("pause");

break;

}

case 6: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

char stu[EVERY];

ElemType s;

ElemType \*e = &s;

cout << "请输入目标学生的学号:" << endl;

cin >> stu;

if (GetElem(Q, stu, e)) {

cout << "学号: " << stu << " 学生的信息为:\n"

<< "ID:" << (\*e).id << "\n"

<< "Name:" << (\*e).name << "\n"

<< "Score:" << (\*e).score << endl;

} else {

cout << "查无此人" << endl;

}

system("pause");

break;

}

case 7:{

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

ElemType s;

cout << "请输入定位学生的学号:" << endl;

cin >> s.id;

int a = 0;

int &b = a;

if (LocateElem(Q,s,b))

cout <<"目标学生成绩表中的位序为" << b <<endl;

else

cout << "查无此人" << endl;

system("pause");

break;

}

case 8: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

cout << "输入成绩表中被查找学生的下面学生的学号" << endl;

ElemType cur\_e;

ElemType s;

ElemType \*pre\_e = &s;

cin >> cur\_e.id;

if (PriorElem(Q, cur\_e, pre\_e)) {

cout << "存在你所查找的学生:\n"

<< "ID:" << (\*pre\_e).id << "\n"

<< "Name:" << (\*pre\_e).name << "\n"

<< "Score:" << (\*pre\_e).score << endl;

} else

cout << "不存在被查找学生" << endl;

system("pause");

break;

}

case 9: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

cout <<"输入成绩表中被查找学生的上面学生的学号" << endl;

ElemType cur\_e;

ElemType s;

ElemType \*next\_e = &s;

cin >> cur\_e.id;

if (NextElem(Q, cur\_e, next\_e)) {

cout << "存在你所查找的学生:\n"

<< "ID:" << (\*next\_e).id << "\n"

<< "Name:" << (\*next\_e).name << "\n"

<< "Score:" << (\*next\_e).score << endl;

} else

cout << "不存在被查找学生" << endl;

system("pause");

break;

}

case 10: {

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

cout << "输入你想插入的位置：" << endl;

int i;

cin >> i;

cout << "输入你想插入的学生的学生信息：" << endl;

ElemType s;

cin >> s.id >> s.name >> s.score;

if (ListInsert(L, i, s))

cout << "插入成功!" << endl;

else

cout << "输入的位序不合法，请重新输入" << endl;

system("pause");

break;

}

case 11:{

if (Isinit != 1) {

cout << "线性表未创建！" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

cout << "请输入你想删除的学生的学号：" << endl;

ElemType s;

cin >> s.id;

ElemType ds;

ElemType \*e1 = &ds;

if (ListDelete(L, s, e1)) {

cout << "你删除的学生的信息为："

<< "ID:" << (\*e1).id << "\n"

<< "Name:" << (\*e1).name << "\n"

<< "Score:" << (\*e1).score << endl;

cout << "删除成功！" << endl;

} else

cout << "不存在此学生" << endl;

system("pause");

break;

}

case 12: {

if (Isinit != 1) {

cout << "线性表未创建!" << endl;

cout << "请重新选择你的操作!" << endl;

system("pause");

break;

}

if (!ListTrabverse(\*L))

cout << "线性表是空表！" << endl;

system("pause");

break;

}

case 13: {

cout << "请输入你想读取的文件名" << endl;

ReadFromFile(L,moro,&Isinit,hw,wh);//表已经被初始化

system("pause");

break;

}

case 14: {

if(moro == 0 && Isinit == 1){

cout << "是否保存(若不保存，则单表中的内容将被清空)当前的单表? 是：1 否：0" << endl;

int i;

cin >> i;

if (i) {

WriteToFile(L);

} else

DestroyList(L);

}

if(moro == 1){

cout << "当前已创建多表，无需再创建!" << endl;

system("pause");

break;

}

cout << "请输入同时操作表的数目" << endl;

int a;

cin >> hm;

if(hm > 0) {

moro = 1;

multilist = (SqList \* )malloc( hm \* sizeof(SqList));

hw = (int \*)malloc( hm \* sizeof(int));

for (a = 0; a < hm; a++) {

hw[a] = 0;

}

wh = 1;

L = &(multilist[0]);

Isinit = hw[0];

cout << "当前操作的表为表 1" << endl;

}else{

cout << "输入的数目不合法，将退出本功能！" << endl;

}

system("pause");

break;

}

case 15: {

if( hm == 0){

cout << "不存在多表！" << endl;

system("pause");

break;

}

//切换表的时候询问上一个表是否保存，若不保存，后面退出的时候可以选择保存

if(moro == 1)

Isinit = hw[wh - 1];

if(Isinit == 1){

cout << "是否要保存(若不保存，退出程序时可再次选择是否保存)上一个表?是：1 否：0" << endl;

int i;

cin >> i;

if (i) {//若文件已经被保存，则切换表的时候无法再次切换到该表

char filename[20];

cout << "输入文件名称（不超过20个字符）:" << endl;

cin >> filename;

strcat(filename,".txt");

ofstream outile;

outile.open(filename,std::ios::out);

if(outile){

if(moro == 1)

hw[wh -1] = 2;

int number;

for(number = 0;number < (\*L).length;number++) {

outile << (\*L).elem[number].id << endl;

outile << (\*L).elem[number].name << endl;

outile << (\*L).elem[number].score << endl;

}

outile.close();

} else

cout << "创建文件失败!" << endl;

}

}

cout << "请输入被切换到表的序号(输入数字0退出切表功能)" << endl;

int a;

while(true){

cin >> a;

if( a == 0)

break;

if( hw[a - 1] == 2 && a <= hm && a > 0){

cout << "此表已被保存！无法切换到该表！请重新输入！" << endl;

}

if( hw[a - 1] == 3 && a <= hm && a > 0){

cout << "此表已被舍弃！无法切换到该表！请重新输入！" << endl;

}

if(a > 0 && a <= hm && hw[a - 1] != 2) {

wh = a;

Isinit = hw[wh - 1];

L = &(multilist[wh - 1]);

break;

}

cout << "输入的表序不存在，请在1--" << hm << "选择输入" << endl;

}

system("pause");

break;

}

case 16: {

int i;

cout << "是否要保存当前表?是：1 否：0" << endl;

cin >> i;

if (i) {

WriteToFile(L);

}

system("pause");

break;

}

case 0: {

int i;

if(moro == 1)//让Isinit为多表中被操作表的状态

Isinit = hw[wh - 1];

if (Isinit != 1) {

if(Isinit == 2)

cout << "此表已保存！" << endl;

else if(Isinit == 0){

cout << "线性表未初始化！" << endl;

cout << "即将退出本功能!" << endl;

if(moro == 1)

hw[ wh -1 ] = 3;

}

}

else{

cout << "是否要保存当前表?是：1 否：0" << endl;

cin >> i;

if (i) {

if (moro == 1)

hw[wh - 1] = 2;

WriteToFile(L);

} else{

//将表改变不保存的状态，即删除该表的存在

if (moro == 1)

hw[wh - 1] = 3;

}

}

//判断是否所有的表被保存，

//若没有，则切换到未保存的表。

if(moro == 1) {

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

if(hw[i] == 1 || hw[i] == 0){

Isinit = hw[i];

L = &multilist[i];

wh = i + 1;

choice = 17;//只是为了再次进入循环，而不是具体的操作命令

break;

}

}

if(choice == 0){

cout << "剩余的表均被舍弃(退出)，即将退出程序！" << endl;

}

}

system("pause");

break;

}

default:

cout << "指令不存在，请重新输入！" << endl;

}//end of switch

}//end of while

cout << "欢迎下次再使用本系统!" << endl;

return 1;

}//end of main()

status IntiaList(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) {

if ((\*L).elem != nullptr) {

free((\*L).elem);

return TRUE;

} else {

return ERROR;

}

}

status ClearList(SqList \*L) {

(\*L).length = 0;

return TRUE;

}

status ListEmpty(SqList L) {

if (L.length == 0)

return TRUE;

else

return FALSE;

}

int ListLength(SqList L) {

return L.length;

}

status GetElem(SqList L, char \*stu, ElemType \*e) {

int a;

for(a = 0;a < L.length;a++) {

if (strcmp(stu,L.elem[a].id) == 0) {

\*e = L.elem[a];

return OK;

}

}

return FALSE;

}

status LocateElem(SqList L, ElemType e,int &a) {

int i;

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

if (strcmp(L.elem[i].id,e.id) == 0) {

a = i + 1;

return TRUE;

}

}

return FALSE;

}

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

int i;

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

if (strcmp(L.elem[i].id,cur\_e.id) == 0 && i != 0) {

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

return TRUE;

}

}

return FALSE;

}

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

int i;

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

if (strcmp(L.elem[i].id,cur\_e.id) == 0 && i != (L.length - 1)) {

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

return TRUE;

}

}

return ERROR;

}

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

if (i < 1 || i > (\*L).length + 1)

return ERROR;

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;

}

ElemType \*q, \*p;

q = &((\*L).elem[i - 1]);

p = &((\*L).elem[(\*L).length - 1]);

for (; p >= q; --p)

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

\*q = e;

++((\*L).length);

return OK;

}

status ListDelete(SqList \*L, ElemType st, ElemType \*e) {

int i;

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

if(strcmp(st.id,(\*L).elem[i].id) == 0)

break;

}

if((\*L).length == i)

return FALSE;

ElemType \*p, \*q;

p = &((\*L).elem[i]);

\*e = \*p;

q = &((\*L).elem[(\*L).length - 1]);

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

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

--((\*L).length);

return OK;

}

status ListTrabverse(SqList L) {

if (!L.elem || L.length == 0)

return ERROR;

int i;

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

cout << "ID:"<< L.elem[i].id << "\n"

<< "Name:" << L.elem[i].name << "\n"

<< "Score:" << L.elem[i].score << "\n" << endl;

}

return TRUE;

}

status ReadFromFile(SqList \*L, int moro, int \*Isinit, int \*hw, int wh) {

char filename[20];

cin >> filename;

strcat(filename, ".txt");

ifstream infile;

infile.open(filename, std::ios::in);

if (infile) {

//根据是多表操作还是单表操作来改变它们的状态，

//从而在前面判断是否被初始化

if(moro == 1)

\*Isinit = hw[wh - 1];

if(\*Isinit == 1){

ClearList(L);

}

if(\*Isinit == 0 && moro == 1) {

IntiaList(L);

\*Isinit = hw[wh - 1] = 1;

} else if(\*Isinit == 0 && moro == 0){

IntiaList(L);

\*Isinit = 1;

}

int a;

for (a = 0;; a++) {

infile >> (\*L).elem[a].id >> (\*L).elem[a].name >> (\*L).elem[a].score;

if(infile.peek() == EOF )

break;

++((\*L).length);

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;

}

}

} else {

cout << "不存在此文件!" << endl;

}

infile.close();

return TRUE;

}

status WriteToFile(SqList \*L) {

char filename[20];

cout << "输入文件名称（不超过20个字符）:" << endl;

cin >> filename;

strcat(filename,".txt");

ofstream outile;

outile.open(filename,std::ios::out);

if(outile){

int number;

for(number = 0;number < (\*L).length;number++) {

outile << (\*L).elem[number].id << endl;

outile << (\*L).elem[number].name << endl;

outile << (\*L).elem[number].score << endl;

}

outile.close();

} else

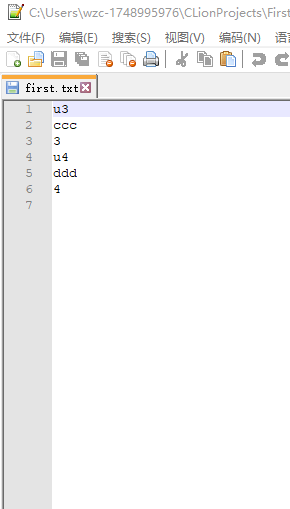
cout << "创建文件失败!" << endl;

return TRUE;

}

### 测试样例：

在文件夹中有first.txt文件，文件的内容如下：



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：IntiaList

测试步骤及结果如表1-1所示

表1-1 IntiaList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  2.按提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3. | 1.主界面输入12进入函数 | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” |

1. 测试函数：DestroyList

测试步骤及结果如表1-2所示

表1-2 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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 主界面输入2进入函数 | 输出“销毁顺序表成功”！ | 输出“销毁顺序表成功”！ |
| 4 | 主界面输入4进入函数 | 输出“线性表未创建！请重新选择你的操作！” | 输出“线性表未创建！请重新选择你的操作！” |

1. 测试函数：ClearList

测试步骤及结果如表1-3所示

表1-3 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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 主界面输入3进入函数 | 输出“清空顺序表成功！” | 输出“清空顺序表成功！” |
| 4 | 主界面输入4进入函数 | 输出“顺序表为空” | 输出“顺序表为空” |

1. 测试函数：ListEmpty

测试步骤及结果如表1-4所示

表1-4 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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 主界面输入4进入函数 | 输出“顺序表不为空” | 输出“顺序表不为空” |
| 4 | 主界面输入3进入函数 | 输出“清空顺序表成功！” | 输出“清空顺序表成功！” |
| 5 | 主界面输入4进入函数 | 输出“顺序表为空” | 输出“顺序表为空” |

1. 测试函数：ListLength

测试步骤及结果如表1-5所示

表1-5 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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 主界面输入5进入函数 | 输出“顺序表的长度为3” | 输出“顺序表的长度为3” |

1. 测试函数：GetElem

测试步骤及结果如表1-6所示

表1-6 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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 1.主界面输入6进入函数  2.按提示输入要被查找学生的学号u1 | 输出“学号: u1 学生的信息为:  ID:u1  Name:aaa  Score:1” | 输出“学号: u1 学生的信息为:  ID:u1  Name:aaa  Score:1” |

1. 测试函数：LocateElem

测试步骤及结果如表1-7所示

表1-7 LocateElem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初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 1.主界面输入7进入函数  2.按提示输入要被查找学生的学号u1 | 输出“目标学生成绩表中的位序为1” | 输出“目标学生成绩表中的位序为1” |

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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 1.主界面输入8进入函数  2.按提示输入顺序表中被查找学生的下面学生的学号u2 | 输出“存在你所查找的学生:  ID:u1  Name:aaa  Score:1” | 输出“存在你所查找的学生:  ID:u1  Name:aaa  Score:1” |

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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 1.主界面输入9进入函数  2.按提示输入顺序表中被查找学生的上面学生的学号u1 | 输出“存在你所查找的学生:  ID:u2  Name:bbb  Score:2” | 输出“存在你所查找的学生:  ID:u2  Name:bbb  Score:2” |

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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 1.主界面输入10进入函数  2.按提示输入你想插入的位置：3. 输入你想插入的学生的学生信息：  u3 ccc 3 | 输出“插入成功！” | 输出“插入成功！” |
| 4 | 主界面输入12进入函数 | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2  ID:u3  Name:ccc  Score:3” | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2  ID:u3  Name:ccc  Score:3” |

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1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 1.主界面输入11进入函数  2.按提示输入被删除学生的学号u1 | 输出“你删除的学生的信息为：ID:u1  Name:aaa  Score:1  删除成功！” | 输出“你删除的学生的信息为：ID:u1  Name:aaa  Score:1  删除成功！” |
| 4 | 主界面输入12进入函数 | 输出“ID:u2  Name:bbb  Score:2” | 输出“ID:u2  Name:bbb  Score:2” |

1. 测试函数：ListTrabverse

测试步骤及结果如表1-12所示

表1-12 ListTrabverse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初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 1.主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3 | 1.主界面输入12进入函数 | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” |

1. 测试函数：ReadFromFile

测试步骤及结果如表1-13所示

表1-13 ReadFromFile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  2.按提示输入要读取的txt文件first | 无输出，文件内容已被读取到当前操作的表中 | 无输出，文件内容已被读取到当前操作的表中 |
| 2 | 主界面输入12进入函数 | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” |

1. 测试函数：MultiList

测试步骤及结果如表1-14所示

表1-14 MultiList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14进入函数  2.按提示输入要操作的线性表数目，输入2 | 输出“操作成功！当前操作的表为表 1” | 输出“操作成功！当前操作的表为表 1” |
| 2 | 1.主界面输入13进入函数  2.按提示输入要读取的txt文件first | 无输出，文件内容已被读取到当前操作的表中 | 无输出，文件内容已被读取到当前操作的表中 |
| 3 | 1.主界面输入12进入函数 | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” |
| 4 | 1.主界面输入15进入函数  2. 根据提示是否要保存(若不保存，退出程序时可再次选择是否保存)上一个表?是：1 否：0  0  请输入被切换到表的序号(输入数字0退出切表功能)  2 | 输出“当前操作的为表 2” | 输出“当前操作的为表 2” |
| 5. | 1.主界面输入1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 6. | 1.主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 7. | 1.主界面输入12进入函数 | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” |

15）测试函数：SwitchList

测试步骤及结果如表1-15所示

表1-15 SwitchList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14进入函数  2.按提示输入要操作的线性表数目，输入2 | 输出“操作成功！当前操作的表为表 1” | 输出“操作成功！当前操作的表为表 1” |
| 2 | 1.主界面输入13进入函数  2.按提示输入要读取的txt文件first | 无输出，文件内容已被读取到当前操作的表中 | 无输出，文件内容已被读取到当前操作的表中 |
| 3 | 1.主界面输入12进入函数 | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” |
| 4 | 1.主界面输入15进入函数  2. 根据提示是否要保存(若不保存，退出程序时可再次选择是否保存)上一个表?是：1 否：0  0  请输入被切换到表的序号(输入数字0退出切表功能)  2 | 输出“当前操作的为表 2” | 输出“当前操作的为表 2” |
| 5. | 1.主界面输入1初始化表  2.按照提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 6. | 1.主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 7. | 1.主界面输入12进入函数 | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” |
| 8. | 1.主界面输入15进入函数  2. 根据提示是否要保存(若不保存，退出程序时可再次选择是否保存)上一个表?是：1 否：0  0  请输入被切换到表的序号(输入数字0退出切表功能)  1 | 输出“当前操作的为表 1” | 输出“当前操作的为表 1” |
| 9. | 1.主界面输入12进入函数 | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” | 输出“ID:u3  Name:ccc  Score:3  ID:u4  Name:ddd  Score:4” |

16）测试函数：SaveList

测试步骤及结果如表1-16所示

表1-16 SaveList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  2.按提示向表中输入要存储的内容 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 | 输出“当前已初始化表!”并且提示输入学生的信息，学生的数目不能超过100个 |
| 2 | 主界面输入学生信息：u1 aaa 1  u2 bbb 2 u3 ccc 3 # | 无输出，学生信息存储在表中 | 无输出，学生信息存储在表中 |
| 3. | 1.主界面输入12进入函数 | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” |
| 4. | 1.主界面输入16进入函数  2.根据提示输入：  是否要保存当前表?是：1 否：0  1  输入文件名称（不超过20个字符）:  second | 无输出，顺序表内容保存在second.txt文件中 | 无输出，顺序表内容保存在second.txt文件中 |
| 5. | 主界面输入2进入函数 | 输出“销毁顺序表成功”！ | 输出“销毁顺序表成功”！ |
| 6. | 1.主界面输入13进入函数  2.按提示输入要读取的txt文件second | 无输出，文件内容已被读取到当前操作的表中 | 无输出，文件内容已被读取到当前操作的表中 |
| 7. | 1.主界面输入12进入函数 | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” | 输出“ID:u1  Name:aaa  Score:1  ID:u2  Name:bbb  Score:2” |

## 实验小结

在本次实验中，让我对顺序存储结构有了更深刻的了解。当进行单表操作设计的时候，并没有遇到比较棘手的问题。但是，在对多表操作设计的时候，遇到了比较难的问题。怎么样来判断不同的表的状态，是已经被初始化，被保存，还是被舍弃。后来想到可以用一个数组用数字来存储表的不同的状态。之前想用引用来对表进行操作，后来发现到多表操作无法进行，还是指针好用。实验之前，最好设计一个表格，来详细地表现出系统的结构，这样便能够高效地设计程序。