

**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**

# 2 基于链式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物理结构采用单链表,熟练掌握线性表的基本运算的实现。

### 实验内容：

采用单链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单链表的方式实现，即线性表中的相邻元素在物理内存的位置上不一定相邻，但数据元素的前后关系不变。

## 系统设计

系统模块构成如图 1-1 所示

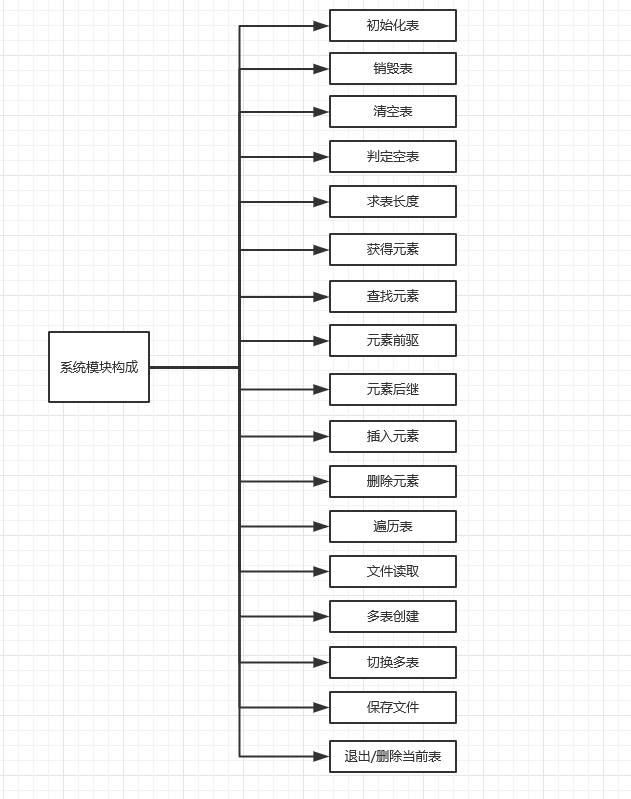


图1-1 系统模块构成图

各个模块的输入、输出以及功能实现介绍如下：   
1.初始化表：

条件一：线性表L不存在

输入：根据提示输入学生的学号、姓名以及成绩

输出：线性表创建成功！

功能实现：创建一个存储学生信息的线性表

条件二：线性表L存在

输入：无

输出：线性表创建失败！

功能实现：重新选择你的操作！

2.销毁表：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：无

输出：销毁线性表成功！

功能实现：销毁当前的线性表，线性表成为未初始化(未创建)状态！  
3.清空表：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：无

输出：清空线性表成功！

功能实现：清空当前的线性表  
4.判定空表：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在且为空

输入：无

输出：线性表为空

功能实现：判定当前线性表是否为空表

条件三：线性表存在且不为空

输入：无

输出：线性表不为空

功能实现：判定当前线性表是否为空表  
5.求表长：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：无

输出：线性表的长度为\*\*

功能实现：输出当前线性表的长度，即包含多少个数据元素   
6.获得元素：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：学生学号

输出：若目标学生存在，打印学生的学生信息；若学生不存在，则打印“查无此人”

功能实现：查找学生并打印学生的信息  
7.查找元素：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：学生学号

输出：若目标学生存在，打印目标学生在表中的位序号；若学生不存在，则打印“查无此人”

功能实现：查找学生并打印学生的位序   
8.获得前驱：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：学生学号

输出：若学生(元素)前驱存在，则打印前驱结点的学生信息；若学生(元

素)前驱不存在，则打印“查无此人”

功能实现：获得并打印目标学生的前驱信息   
9.获得后继：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：学生学号

输出：若学生(元素)后继存在，则打印后继结点的学生信息；若学生(元

素)后继不存在，则打印“查无此人”

功能实现：获得并打印目标学生的后继信息  
10.插入元素：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：输入要插入的位序以及学生的信息

输出：若位序合法，打印“插入成功”；若位序不合法，则打印“输入的位序不合法”

功能实现：在合法的位置插入学生信息   
11.删除元素：条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：输入要被删除学生的学号

输出：若学生存在，则打印被删除学生的信息；否则，打印“目标学生不存在”

功能实现：删除表中已存在的学生的信息   
12.遍历表：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：无

输出：按次序输出表中学生的信息

功能实现：遍历并输出表中学生的信息  
13．读取文件：

条件一：线性表L不存在

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：线性表L存在

输入：输入要读取的文件名称

输出：若文件存在并打开成功，打印“读取成功”；否则，打印“不存在此文件”

功能实现：将已存在的文件读取到线性表中  
14.创建多表：

条件一：线性表L不存在

输入：输入你要创建的多表数量

输出：多表创建成功！当前操作的表位表1

功能实现：创建指定数量的多表

条件二：线性表L存在

输入：根据提示是否保存当前单表。若保存，输入1后输入文件名称；否则，输入0后操作情况与条件一相同

输出：创建文件成功!(保存)

功能实现：询问是否保存单表并创建多表

条件三：多表已存在

输入：无

输出：当前已创建多表，无需再次创建

功能实现：拒绝重复创建多表   
15.切换多表：

条件一：多表不存在

输入：无

输出：不存在多表！

功能实现：单表无法切换表

条件二：多表存在且目标表存在

输入：输入目标表的序号

输出：当前操作的表为表\*

功能实现：切换到指定的表

条件三：多表存在且目标表不存在或已被删除

输入：输入目标表的序号

输出：此表已被舍弃！无法切换到该表！请重新输入！

功能实现：拒绝切换到被舍弃(删除)的表

条件四：多表已存在且输入的表的序号不存在

输入：输入目标表的序号

输出：输入的表序不合法，请重新输入

功能实现：拒绝切换到不存在的表   
16.保存文件：

条件一：当前表未创建

输入：无

输出：线性表未创建！请重新选择你的操作！

功能实现：重新选择你的操作

条件二：当前表已创建

输入：输入保存的文件名称

输出：创建文件成功!

功能实现：将表中的内容保存到指定文件中

功能实现：拒绝切换到不存在的表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1-2所示）

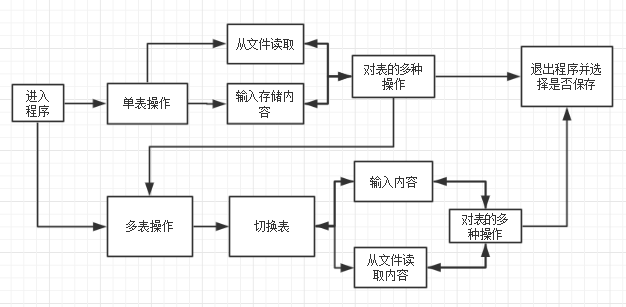


图1-2 程序的系统设计流程图

## 系统实现

### 系统各模块算法实现以及函数调用

1.初始化表：引用的函数名称是InitaList(\*L)

在函数中为表初始化，即为第一个元素结点分配空间，第一个结点不存储任何信息；之后根据用户输入来不断地为后面的结点分配空间。当用户的输入结束时，系统停止分配空间并将尾结点的指针设置为空。之后在模块中将表的状态设置为已创建的状态。

2.销毁表：引用的函数是DestroyList(\*L)

在函数中从首结点遍历到尾结点，将各个结点所占的空间逐一释放掉。之后在模块中将表的状态设置为未创建的状态。

3.清空表：引用的函数是ClearList(\*L)

在函数中将表的长度设置为0，模块中不做任何操作。

4.判定空表：引用的函数是ListEmpty(L)

在函数判断表的长度是否为0，若表的长度为0，当前表判定为空表；否则，当前表位非空表。模块中不做任何操作。

5.求表长：引用的函数是ListLength(L)

在函数中返回当前表的长度并且在模块中输出函数的返回值，从而得到当前操作表的长度。

6.获得元素：引用的函数是GetElem(L,stu,e)

在模块中首先创建一个Elemtype类型的变量e和一个字符类型指针stu。e中用来存储函数中查询到的学生信息，stu用来存储用户输入的学生学号。在函数中逐个遍历所有的结点，若结点中的存在学生的id与stu相同，则将学生的信息复制到e中，函数返回OK并在模块中输出学生的信息；否则，函数返回FALSE并在模块中输出告知用户学生不存在的信息。

7.查找元素：引用的函数是LocateElem(L,e,a)

在模块中首先创建一个Elemtype类型的变量e和一个整数类型a。e中的id用来存储用户查询学生的id ，a用来存储函数返回的学生位序。函数中逐个遍历所有的结点，若结点中的存在学生的id与e中的id信息相同，将学生的位序复制到a中，函数返回OK并在模块中输出学生的位序；否则，函数返回FALSE并在模块中输出告知用户学生不存在的信息。

8.获得前驱：引用的函数是PriorElem(L,cur\_e,pre\_e)

cur\_e中的id用来存储用户输入的学生学号，pre\_e用来存储函数中查找结点的前驱信息。函数中逐个遍历所有的结点，若表中存在结点中的id与cur\_e中的id相同，则将学生的信息复制到pre\_e中，函数返回OK并在模块中输出前驱结点学生的信息；否则，函数函数返回FALSE并在模块中输出告知用户学生不存在的信息。

9.获得后继：引用的函数是NextElem(L,cur\_e,next\_e)

cur\_e中的id用来存储用户输入的学生学号，next\_e用来存储函数中查找结点的后继信息。函数中逐个遍历所有的结点，若表中存在结点中的id与cur\_e中的id相同，则将学生的信息复制到next\_e中，函数返回OK并在模块中输出后继结点学生的信息；否则，函数函数返回FALSE并在模块中输出告知用户学生不存在的信息。

10.插入元素：引用的函数是ListInsert(\*L,i,e)

e用来存储用户要插入的学生信息，i用来存储用户输入的要插入的位置。若该位置合法(存在),则在该位置插入学生的信息，并且在将其前面的结点指针设置为指向新结点，将新结点的指针指向前驱结点的指针原来指向的位置，函数返回OK；否则，函数返回FALSE并且模块中输出告知用户位置不合法的信息。

11.删除元素：引用的函数是ListDelete(\*L,i,e)

e用来存储用户要插入的学生信息，i用来存储用户输入的要删除结点的位置。若该位置合法(存在),则该位置学生的信息，并且在将其前面的结点指针设置为指向其后面的结点，函数返回OK；否则，函数返回FALSE并且模块中输出告知用户位置不合法的信息

12.遍历表：引用的函数是ListTraverse(L)

在函数中遍历表中所有的结点并且输出结点的信息，若结点的指针指向为空，则遍历结束，函数返回OK。

13.读取文件：引用的函数是ReadFromFile(\*L,moro,\*Isinit,\*hw,wh)

moro用来判断当前表是否为多表中一表，Isinit用来判断当前表是否被初始化，hw用来读取多表中某表的状态，wh用来判断当前在多表中的位置(若该表为多表中的一表)。经过判断后，若当前表已将创建，则询问用户是否保存当前表的信息，若保存，则调用WriteToFile(\*L)；否则，输入要读取的文件名。读取文件时，若当前表已创建，则调用DestroyList(\*L)。然后从文件中逐个读取信息，读取一个信息就分配一个结点的空间来存储结点的信息，直到读取到文件尾。

14.创建多表：无具体函数

在模块中创建多个表指针来指向不同的表。创建表后，当前操作的表默认为表1。

15.切换多表：无具体函数

在切换之前，首先询问用户是否保存当前表。若保存，则调用WriteToFile(\*L);若不保存，则程序继续运行。在模块中首先判断要切换的表是否存在，若已经被删除，则不能切换到该表并且输出告知用户该表已被删除的信息。若该表存在，则切换到该表。

16.保存文件：引用的函数是WriteToFile(\*L)

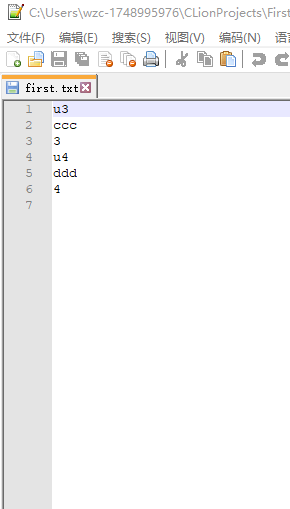
在函数中要求用户输入文件名，之后遍历表中所有的结点并且将结点中的信息打印到文件中。若文件已存在，则会覆盖原文件的内容。

0.退出：无引用函数

经过和13读取文件 开始部分相同的判断，若当前表为单表，则询问用户是否保存。若保存，则调用WriteToFile(\*L);否则，将退出程序。若当前表为多表中的一表，则询问用户是否保存。若不保存，则删除该表并且自动切换到多表中未删除的表。

### 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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“销毁线性表成功”！ | 输出“销毁线性表成功”！ |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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“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 | 输出“清空线性表成功！” | 输出“清空线性表成功！” |
| 5 | 主界面输入4进入函数 | 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1-5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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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“线性表的长度为3” | 输出“线性表的长度为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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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 | 输出“当前已初始化表!”并且提示输入学生的信息 |
| 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，线性表内容保存在second.txt文件中 | 无输出，线性表内容保存在second.txt文件中 |
| 5. | 主界面输入2进入函数 | 输出“销毁线性表成功”！ | 输出“销毁线性表成功”！ |
| 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，让我对线性存储结构有了更深刻的了解。经过第一次顺序表的数据结构实验，本次实验的和上次实验内容相似，这次实验设计比较顺利，谢谢老师和助教对我的精心指导！

附录

/\*\*

\* 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，第一个结点不存储任何信息

\* 每个学生信息的为链式存储

\*/

#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;

if(moro == 1) {

hw[wh - 1] = 3;

Isinit = hw[wh - 1];

}

else

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;

}