

华中科技大学

课程实验报告

课程名称： 数据结构实验

专业班级 CS2210

学 号 U202115473

姓 名 刘欣逸

指导教师 郑渤龙

报告日期 2023 年 5 月 29 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
1.1	问题描述.....	1
1.2	系统设计.....	2
1.3	系统实现.....	7
1.4	系统测试.....	7
1.5	实验小结.....	16
2	基于二叉链表的二叉树实现	18
2.1	问题描述.....	18
2.2	系统设计.....	18
2.3	系统实现.....	18
2.4	系统测试.....	18
2.5	实验小结.....	18
3	课程的收获和建议	19
3.1	基于顺序存储结构的线性表实现	19
3.2	基于链式存储结构的线性表实现	19
3.3	基于二叉链表的二叉树实现.....	19
3.4	基于邻接表的图实现.....	19
4	附录 A 基于顺序存储结构线性表实现的源程序	19
5	附录 B 基于链式存储结构线性表实现的源程序.....	19
6	附录 C 基于二叉链表二叉树实现的源程序	19
7	附录 D 基于邻接表图实现的源程序.....	19

1 基于顺序存储结构的线性表实现

1.1 问题描述

要求构造一个具有菜单的功能演示系统。该演示系统实现多个线性表管理。其中，在主函数中准备函数调用所需实参值、显示函数执行结果，并给出适当的操作提示。

实现依据最小完备性和常用性相结合的原则确定的初始化表、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。

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L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L已存在， $1 \leq i \leq \text{ListLength}(L)$ ；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,compare())；初始条件是线性表L已存在；操作结果是返回L中第一个与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。
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，

$1 \leq i \leq \text{ListLength}(L) + 1$; 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, $1 \leq i \leq \text{ListLength}(L)$; 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, 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()`。

此外, 还在基础功能的基础上实现了附加功能:

1. 最大连续子数组和: 函数名称是`MaxSubArray(L)`; 初始条件是线性表L已存在且非空, 请找出一个具有最大和的连续子数组(子数组最少包含一个元素), 操作结果是其最大和;
2. 和为K的子数组: 函数名称是`SubArrayNum(L, k)`; 初始条件是线性表L已存在且非空, 操作结果是该数组中和为k的连续子数组的个数;
3. 顺序表排序: 函数名称是`SortList(L)`; 初始条件是线性表L已存在; 操作结果是将L由小到大排序;
4. 实现线性表的文件形式保存: 其中,
 - 需要设计文件数据记录格式, 以高效保存线性表数据逻辑结构 $(D, \{R\})$ 的完整信息;
 - 需要设计线性表文件保存和加载操作合理模式。
5. 实现多个线性表管理: 设计相应的数据结构管理多个线性表的查找、添加、移除等功能。

实验目的:

1. 加深对线性表的概念、基本运算的理解;
2. 熟练掌握线性表的逻辑结构与物理结构的关系;
3. 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 系统设计

本系统提供一个采用顺序存储方式的线性表及其操作实现。系统可供选择的操作有:

- 基本操作：初始化线性表、销毁表、清空表、判定空表、求表长、获得元素、查找元素、获得某元素的前驱、获得某元素的后继、插入元素、删除元素、遍历线性表。
- 附加功能：最大连续子数组和、和为K的子数组、顺序表排序、实现线性表的文件形式保存、实现多个线性表管理。

1.2.1 头文件、宏和类型定义

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <limits.h>
5
6  #define TRUE 1
7  #define FALSE 0
8  #define OK 1
9  #define ERROR 0
10 #define INFEASIBLE -1
11 #define OVERFLOW -2
12 #define ISEMPY -3
13 #define LIST_INIT_SIZE 1000
14 #define LISTINCREMENT 10
15 #define max(i,j) ((i)>(j)?(i):(j))
16
17 typedef int status;
18 typedef int ElemType; //数据元素类型定义
19 typedef struct{ //顺序表（顺序结构）的定义
20     ElemType * elem;
21     int length;
22     int listsize;
23 } SqList;
24
```

```
25 typedef struct THELISTS{ //顺序表的管理表定义
26     struct ALIST{
27         char name[30];
28         SqList L;
29     } elem[50];
30     int length = 0;
31     int listssize = 50;
32 } LISTS;
```

1.2.2 函数设计

1. 函数名称: `InitaList(L)`;
初始条件: 线性表L不存在;
操作结果: 是构造一个空的线性表;
算法思路: 先分配存储空间后, 将表长设为0, 再将线性表容量变量设为预定义的初始存储容量。
2. 函数名称: `DestroyList(L)`;
初始条件: 线性表L已存在;
操作结果: 销毁线性表L;
算法思路: 释放内存并将其他结构成员设置为初值。
3. 函数名称: `ClearList(L)`;
初始条件: 线性表L已存在;
操作结果: 将L重置为空表;
算法思路: 将表长设为0。
4. 函数名称: `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;
算法思路: 表长为0则为空表, 否则不是空。
5. 函数名称: `ListLength(L)`;
初始条件: 线性表L已存在;
操作结果: 返回L中数据元素的个数;

算法思路：返回线性表表长的值。

6. 函数名称：GetElem(L,i,e);

初始条件：线性表已存在， $1 \leq i \leq \text{ListLength}(L)$;

操作结果：用e返回L中第i个数据元素的值;

算法思路：将线性表中第i个数据元素的值赋值给e。

7. 函数名称：LocateElem(L,e,compare());

初始条件：线性表已存在;

操作结果：返回线性表中第1个与e相等的数据元素的位置，若这样的数据元素不存在，则返回值为0;

算法思路：从索引低位开始顺序查找，如果找到返回该元素的位序

8. 函数名称：PriorElem(L,cur_e,pre_e);

初始条件：线性表L已存在;

操作结果：若cur_e是L的数据元素并且不是第一个数据元素，用pre_e返回它的前驱，否则操作失败。

算法思路：首先判断该元素不是第一个数据元素，再查找该结点，如果查找成功且该结点有前驱元素，则将前驱元素赋值给pre_e。若未找到该结点，或者该结点是第一个数据元素，则返回ERROR。

9. 函数名称：NextElem(L,cur_e,next_e)

初始条件：线性表L已存在;

操作结果：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。

算法思路：首先判断该元素不是最后一个数据元素，再查找该结点，如果查找成功且该结点有后继元素，则将后继元素赋值给next_e。若未找到该结点，或者该结点是最后一个数据元素，则返回ERROR。

10. 函数名称：ListInsert(L,i,e)

初始条件：线性表L已存在且非空， $1 \leq i \leq \text{ListLength}(L)+1$;

操作结果：在L的第i个位置之前插入新的数据元素e。

算法思路：先遍历顺序表，若线性表L已存在且不为空，输入的i值不合法，则返回ERROR; 若i的值合法，则在线性表的第i个位置前插入新数据元素e，返回OK。若线性表L不存在，返回INFEASIBLE。

11. 函数名称：ListDelete(L,i,e);

初始条件：线性表L已存在且非空， $1 \leq i \leq \text{ListLength}(L)$ ；

操作结果：删除L的第i个数据元素，用e返回其值；

设计思想：先遍历顺序表，如果线性表L已存在且非空，并且输入的i值不合法，则返回ERROR。若满足线性表L已存在且L非空，并且i的值合法，则删除线性表的第i个位置的数据元素，并用e返回其值，返回OK。

12. 函数名称：ListTraverse(L);、

初始条件：线性表L已存在；

操作结果：依次遍历L的每个数据元素；

设计思想：若线性表L存在，则依序遍历元素；否则返回ERROR。

13. 函数名称：SaveList(L,filename);

初始条件：线性表L已存在；

操作结果：将线性表L保存到指定文件；

算法思路：用fprintf保存为文件。

14. 函数名称：LoadList(L);

初始条件：文件filename已存在；

操作结果：从文件中加载数据到L；

算法思路：用fscanf将文件读取顺序表。

15. 函数名称：MaxSubArray(L);

初始条件：线性表L已存在；

操作结果：返回线性表L的最大连续子数组和；

算法思路：逐个枚举子数组求出最大连续子数组和。

16. 函数名称：SubArrayNum(L,k);

初始条件：线性表L已存在；

操作结果：返回线性表L的和为k的连续子数组个数；

算法思路：逐个枚举子数组求出和为k的连续子数组个数。

17. 函数名称：SortList(L)

初始条件：线性表L已存在；

操作结果：将L中的元素按升序排序；

算法思路：用冒泡排序算法将L的数据元素按升序排序。

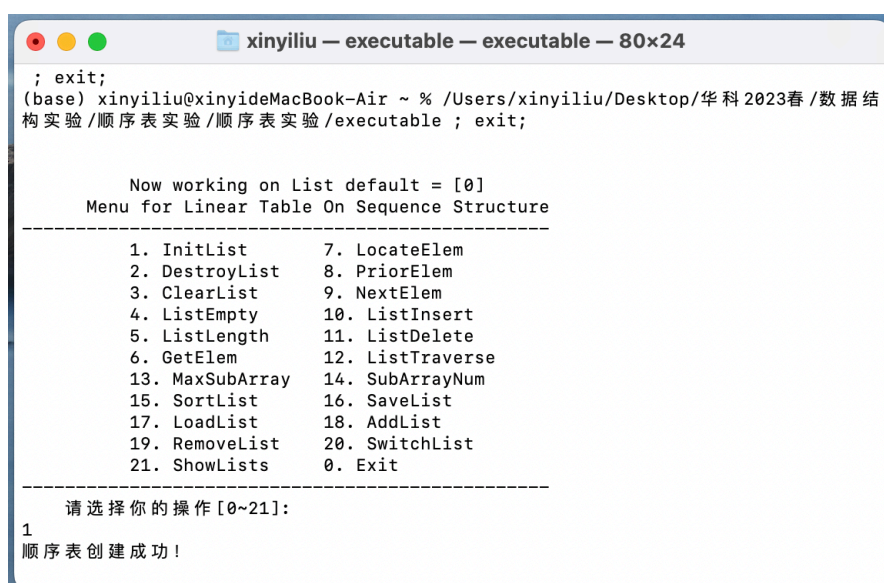
1.3 系统实现

见《附录 A 基于顺序存储结构线性表实现的源程序》。

1.4 系统测试

先用插入操作创建包含元素 1、2、3、4、5、6、7 的顺序表。再测试判空、查找、删除、遍历、销毁等基础功能。

1. InitList()

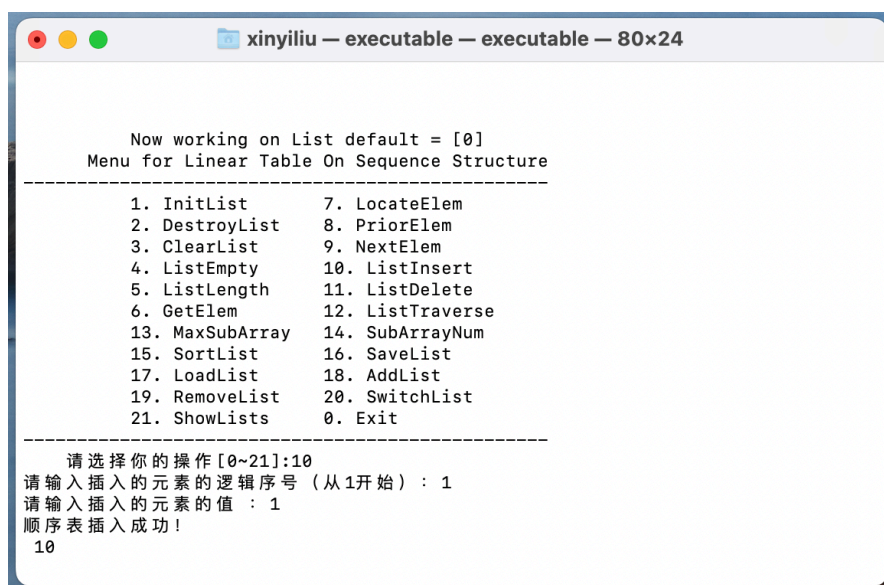


```
xinyiliu — executable — executable — 80x24
; exit;
(base) xinyiliu@xinyideMacBook-Air ~ % /Users/xinyiliu/Desktop/华科2023春/数据结构实验/顺序表实验/顺序表实验/executable ; exit;

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList 20. SwitchList
21. ShowLists  0. Exit
-----

请选择你的操作 [0~21]:
1
顺序表创建成功!
```

2. ListInsert() 依次插入了元素 1、2、3、4、5、6、7。



```
xinyiliu — executable — executable — 80x24

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList 20. SwitchList
21. ShowLists  0. Exit
-----

请选择你的操作 [0~21]:10
请输入插入的元素的逻辑序号 (从1开始): 1
请输入插入的元素的值 : 1
顺序表插入成功!
10
```

3. ListEmpty()

```
Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem     12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists  0. Exit
-----
请选择你的操作[0~21]:4
顺序表不为空!

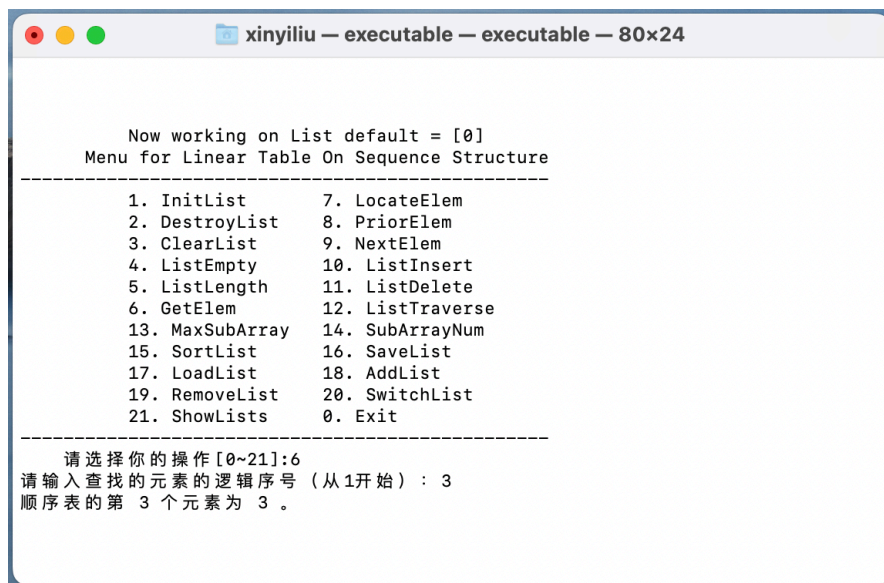
Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
```

4. ListLength()

```
Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem     12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists  0. Exit
-----
请选择你的操作[0~21]:5
顺序表长度为：7。

Now working on List default = [0]
```

5. GetElem()



6. LocateElem()



7. PriorElem()

```
xinyiliu — executable — executable — 80x24

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList 20. SwitchList
21. ShowLists  0. Exit
-----

请选择你的操作[0~21]:8
请输入要查找前驱的元素: 4
元素 4 的前驱是 3。
```

8. NextElem()

```
xinyiliu — executable — executable — 80x24

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList 20. SwitchList
21. ShowLists  0. Exit
-----

请选择你的操作[0~21]:9
请输入要查找后继的元素: 4
元素 4 的后继是 5。
```

9. ListDelete()

```
xinyiliu — executable — executable — 80x24

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists   0. Exit
-----

请选择你的操作[0~21]:11
请输入要删除的元素的逻辑序号: 7
被删除的元素是 7。
```

10. ListTraverse()

```
xinyiliu — executable — executable — 81x24

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists   0. Exit
-----

请选择你的操作[0~21]:12

-----all elements -----
1 2 3 4 5 6
-----end -----

遍历成功！
```


11. ClearList()

```
xinyiliu — executable — executable — 81x24
顺序表插入成功！

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList   8. PriorElem
3. ClearList     9. NextElem
4. ListEmpty     10. ListInsert
5. ListLength    11. ListDelete
6. GetElem       12. ListTraverse
13. MaxSubArray  14. SubArrayNum
15. SortList     16. SaveList
17. LoadList    18. AddList
19. RemoveList   20. SwitchList
21. ShowLists    0. Exit
-----
请选择你的操作[0~21]:3
顺序表清空成功！
```

12. DestroyList()

```
xinyiliu — executable — executable — 81x24
子数组和为 6 的个数为 2 。

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList   8. PriorElem
3. ClearList     9. NextElem
4. ListEmpty     10. ListInsert
5. ListLength    11. ListDelete
6. GetElem       12. ListTraverse
13. MaxSubArray  14. SubArrayNum
15. SortList     16. SaveList
17. LoadList    18. AddList
19. RemoveList   20. SwitchList
21. ShowLists    0. Exit
-----
请选择你的操作[0~21]:2
顺序表删除成功！

Now working on List default = [0]
```

对顺序表 7,6,5,4,3,2,1 测试附加功能:

1. MaxSubArray()



```
Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList          7. LocateElem
2. DestroyList       8. PriorElem
3. ClearList         9. NextElem
4. ListEmpty         10. ListInsert
5. ListLength        11. ListDelete
6. GetElem           12. ListTraverse
13. MaxSubArray       14. SubArrayNum
15. SortList         16. SaveList
17. LoadList         18. AddList
19. RemoveList       20. SwitchList
21. ShowLists        0. Exit
-----
请选择你的操作[0~21]:13
最大子数组和为 28。
```

2. SubArrayNum()



```
Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList          7. LocateElem
2. DestroyList       8. PriorElem
3. ClearList         9. NextElem
4. ListEmpty         10. ListInsert
5. ListLength        11. ListDelete
6. GetElem           12. ListTraverse
13. MaxSubArray       14. SubArrayNum
15. SortList         16. SaveList
17. LoadList         18. AddList
19. RemoveList       20. SwitchList
21. ShowLists        0. Exit
-----
请选择你的操作[0~21]:14
请输入子数组和K: 6
子数组和为 6 的个数为 2。
```

3. SortList()

```
xinyiliu — executable — executable — 81x44

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists  0. Exit
-----

请选择你的操作[0~21]:15
排序操作成功!

Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists  0. Exit
-----

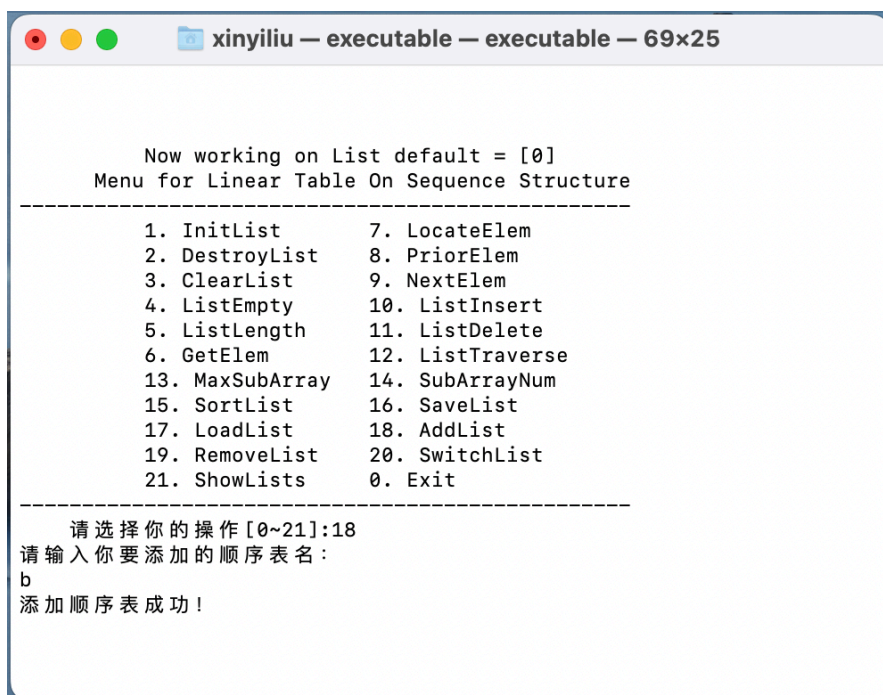
请选择你的操作[0~21]:12
-----all elements -----
1 2 3 4 5 6 7
-----end -----

遍历成功!
```


4. SaveList()、LoadList()、List()等多线性表管理功能。



```
Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists  0. Exit
-----
请选择你的操作[0~21]:16
请输入你要保存的文件名:
file
保存顺序表到文件操作成功!
```



```
Now working on List default = [0]
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList  8. PriorElem
3. ClearList    9. NextElem
4. ListEmpty    10. ListInsert
5. ListLength   11. ListDelete
6. GetElem      12. ListTraverse
13. MaxSubArray 14. SubArrayNum
15. SortList    16. SaveList
17. LoadList   18. AddList
19. RemoveList  20. SwitchList
21. ShowLists  0. Exit
-----
请选择你的操作[0~21]:18
请输入你要添加的顺序表名:
b
添加顺序表成功!
```

```
xinyiliu — executable — executable — 69x25

Now working on List b
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList   8. PriorElem
3. ClearList     9. NextElem
4. ListEmpty     10. ListInsert
5. ListLength    11. ListDelete
6. GetElem       12. ListTraverse
13. MaxSubArray  14. SubArrayNum
15. SortList     16. SaveList
17. LoadList    18. AddList
19. RemoveList   20. SwitchList
21. ShowLists    0. Exit
-----

请选择你的操作[0~21]:20
请输入你要切换到的顺序表名:
b
切换到顺序表 b 成功!

xinyiliu — executable — executable — 69x30

Now working on List b
Menu for Linear Table On Sequence Structure
-----
1. InitList      7. LocateElem
2. DestroyList   8. PriorElem
3. ClearList     9. NextElem
4. ListEmpty     10. ListInsert
5. ListLength    11. ListDelete
6. GetElem       12. ListTraverse
13. MaxSubArray  14. SubArrayNum
15. SortList     16. SaveList
17. LoadList    18. AddList
19. RemoveList   20. SwitchList
21. ShowLists    0. Exit
-----

请选择你的操作[0~21]:17
请输入你要读取的文件名:
file
请输入你要读取到的顺序表名:
b
-----all elements -----
1 2 3 4 5 6 7
-----end -----
从文件读取到顺序表操作成功!
```

1.5 实验小结

本次实验是对我 C 语言、数据结构知识掌握程度的一次检验。在这次实验中我发现了自己的许多不足，如对 C 语言文件读写的掌握不足，代码规范仍有不足。这次实验加强了我对顺序表的理解和掌握，增强了我的编写代码的能力。

最后，感谢老师和助教的指导和帮助。

2 基于二叉链表的二叉树实现

2.1 问题描述

2.2 系统设计

2.3 系统实现

2.4 系统测试

2.5 实验小结

3 课程的收获和建议

3.1 基于顺序存储结构的线性表实现

3.2 基于链式存储结构的线性表实现

3.3 基于二叉链表的二叉树实现

3.4 基于邻接表的图实现

4 附录 A 基于顺序存储结构线性表实现的源程序

5 附录 B 基于链式存储结构线性表实现的源程序

6 附录 C 基于二叉链表二叉树实现的源程序

7 附录 D 基于邻接表图实现的源程序