

**数据结构实验测试用例**

**计算机科学与技术学院**

**2023年春季**

目录

[1基于顺序存储结构的线性表实现 2](#_Toc2161)

[1.1 实验目的 2](#_Toc12779)

[1.2 系统测试用例 2](#_Toc28058)

[2基于链式存储结构的线性表实现 14](#_Toc9257)

[2.1 实验目的 14](#_Toc19926)

[2.2 系统测试用例 14](#_Toc2611)

[3基于二叉链表的二叉树实现 26](#_Toc1187)

[3.1 实验目的 26](#_Toc13231)

[3.2 系统测试用例 26](#_Toc1188)

[4基于邻接表的图实现 36](#_Toc3048)

[4.1 实验目的 36](#_Toc2665)

[4.2 系统测试用例 36](#_Toc2540)

实验说明

2022-2023年度春季数据结构实验采用线上线下结合的方式，每个实验都分为线上和线下两个部分。

线上任务：在educoder上严格按照相关操作的语义，实现各个实验中的函数，确保各函数的正确性；

线下任务：将线上实现的函数组装在一起，实现一个相关抽象数据类型的演示系统。比如“基于二叉链表的二叉树操作演示系统”，实现时既可以实现单棵二叉树操作的演示，也可以实现多棵二叉树的演示。线上任务完成后方可在实验课堂上进行面对面系统功能检查，从而进行综合评分。

线下每个同学最多两次检查机会。助教检查期间根据所提供的测试用例认真检查后，在“《数据结构实验》功能检查记载表”上进行打分。打分规则：每个实验检查表中基本功能一列满分1分（满足所有相关功能测试用例），部分满足按程度酌情扣减（0.9，0.8，...不得低于0.6）；附加功能根据完成情况，1个附加功能0.5分（本文档提供了文件读写和多表等功能的测试用例，其他附加功能需要根据现场情况给出）；代码规范1分（命名、注释、排版等）。总分两列：第一列是前面各功能检查小分相加，第二列是换算成百分制。

线上平台网址：<https://www.educoder.net/>，各位同学实名注册后，使用各自班级的邀请码，以学生身份加入《数据结构实验2022春季》课堂。各班的邀请码：

CS2201：KGVAZS CS2202：MK7BE5

CS2203：HCYWZT CS2204：LW98JC

CS2205：RWXU3Z CS2206：ZX8W5F

CS2207：GNPUDX CS2208：6PDACY

CS2209：3Y5LZQ CS2210：DVEWXJ

CS2211：VQWP9C

计卓2201：48J6KS 本硕博2201：EDR3TU

注：线上提交实验作业均严格按照系统设置时间。

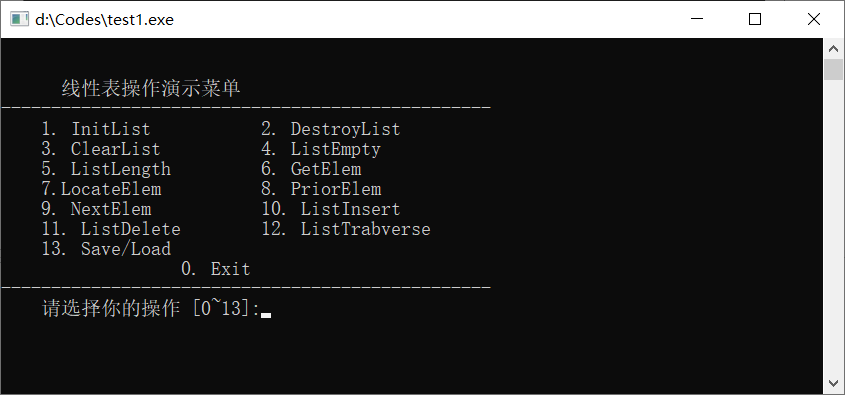
# 1基于顺序存储结构的线性表实现

## 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 系统测试用例

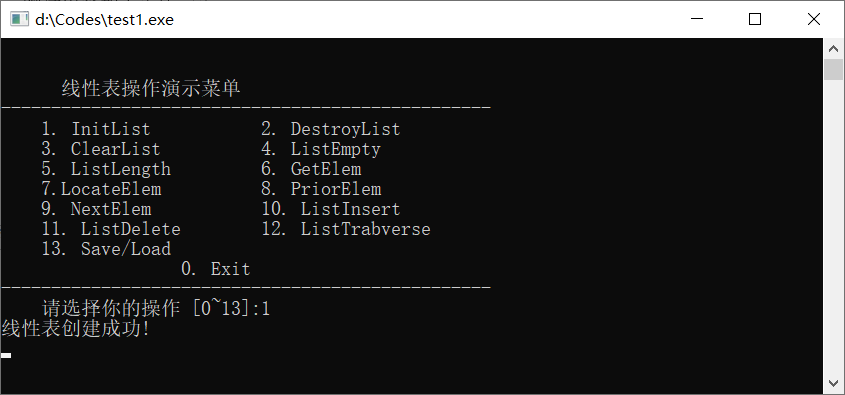
**1.2.1 单线性表操作**



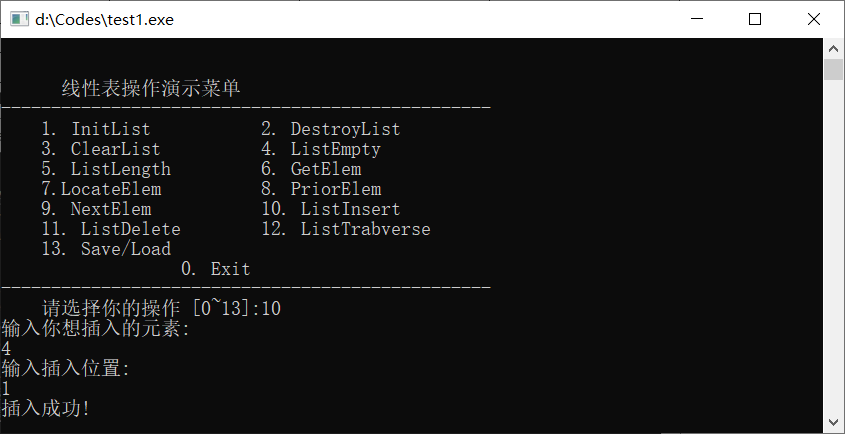
测试内容如下表所示

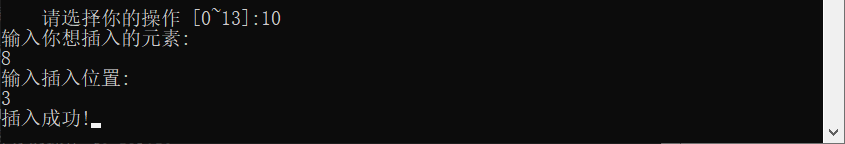
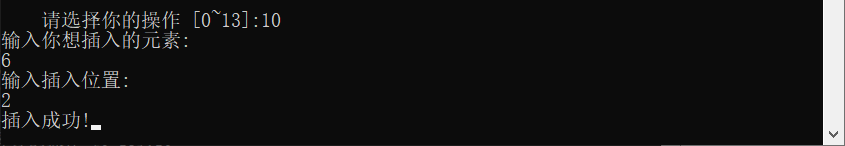
|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能**  **及其对应序号** | **输入** | **预期输出** | **当前操作完成后**  **线性表状态** |
| 1.构造空线性表 | \ | 线性表创建成功! | 空表 |
| 1.构造空线性表 | \ | 线性表创建失败! | 空表，已存在的表无法再次构造 |
| 10.插入元素  （3次） | 1 4；2 6；3 8；  （前者为插入位置，后者为元素值） | 插入成功！（3次） | （3个元素）4 6 8 |
| 4.判空线性表 | \ | 线性表不为空! | 同上 |
| 5.求表长 | \ | 线性表的长度是 3 | 同上 |
| 6.获取元素 | 2（要获取的元素的位置） | 第2个元素是6 | 同上 |
| 7.定位元素 | 8（要定位的元素） | 该数字在第3位 | 同上 |
| 8.获取前驱 | 4（要获取前驱的元素） | 这个元素没有前驱! | 同上 |
|  | 6（要获取前驱的元素） | 前驱是4 | 同上 |
| 9.获取后继 | 8（要获取后继的元素） | 这个元素没有后继! | 同上 |
|  | 4（要获取后继的元素） | 后继是6 | 同上 |
| 12.遍历线性表 | \ | 4 6 8 | 同上 |
| 13.文件保存/  文件读取 | \ | 保存成功!  载入成功! | 同上（将线性表中的数据保存到test.txt中后再从文件中读取数据到线性表中） |
| 11.删除元素 | 2（要删除元素的位置） | 删除成功！ | （2个元素）4 8 |
| 3.清空线性表 | \ | 线性表清空成功! | 空表 |
| 2.销毁线性表 | \ | 线性表销毁成功! | 线性表空间被释放 |
| 2.销毁线性表 | \ | 线性表销毁失败! | 无法销毁两次 |
| 0.退出系统 | \ | 欢迎再次使用本系统！ | 同上 |

**第一步：1.构造空线性表**

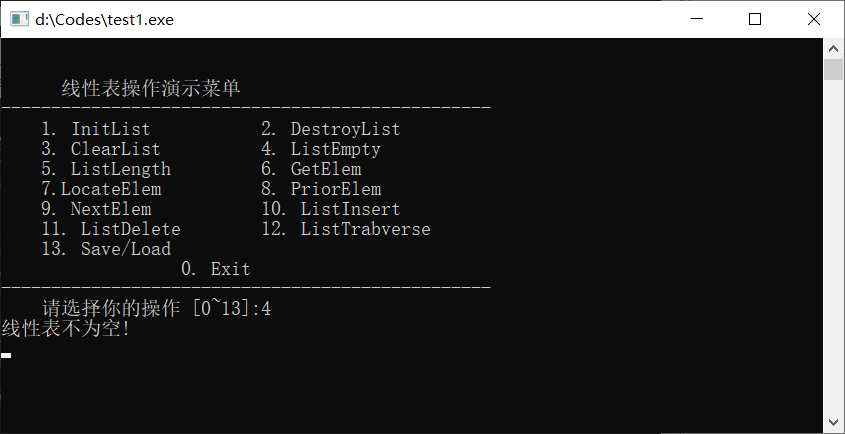


**第二步：10.插入元素（3次）**

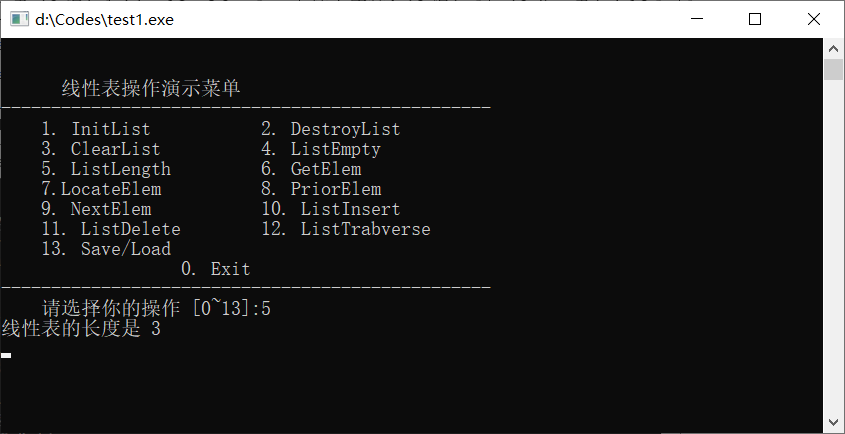




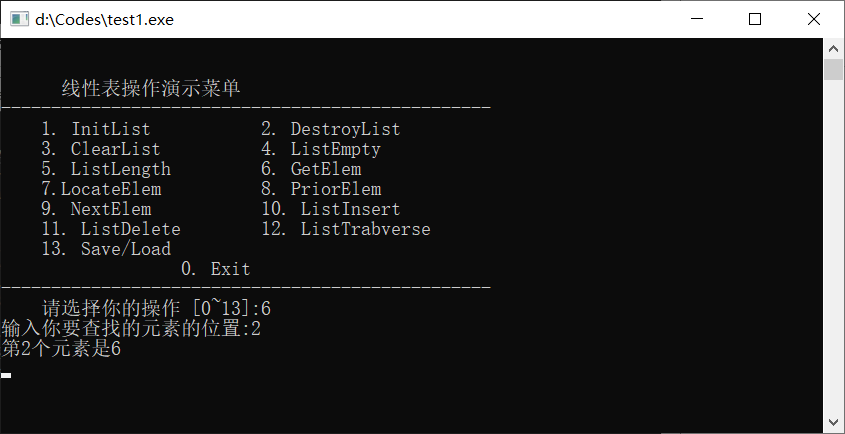
**第三步：4.判空线性表**



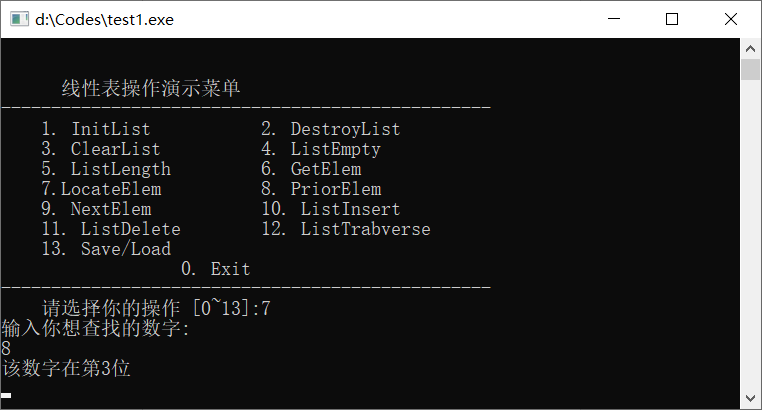
**第四步：5.求表长**



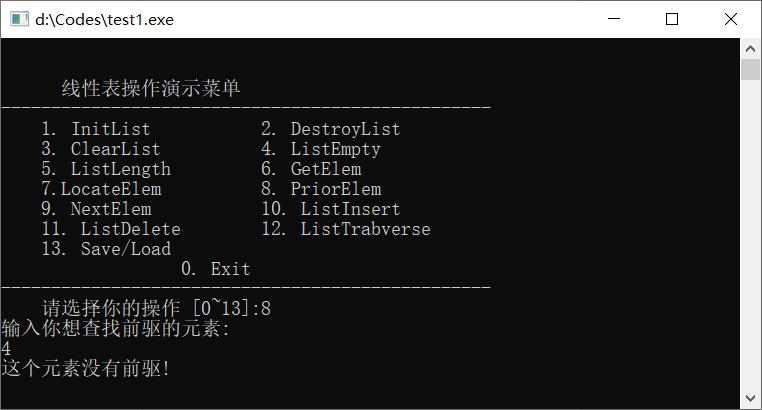
**第五步：6.获取元素**

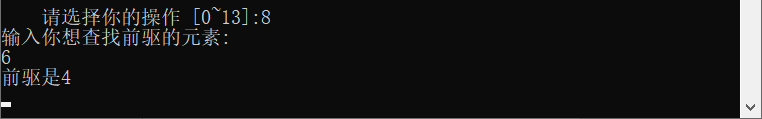


**第六步：7.定位元素**

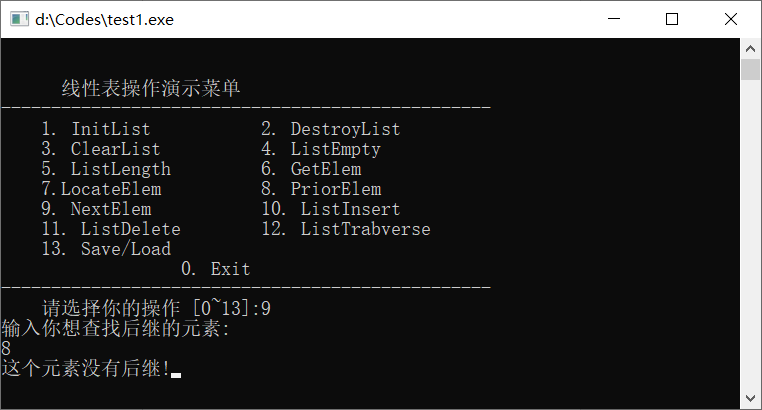


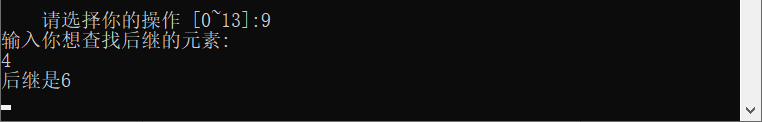
**第七步：8.获取前驱**



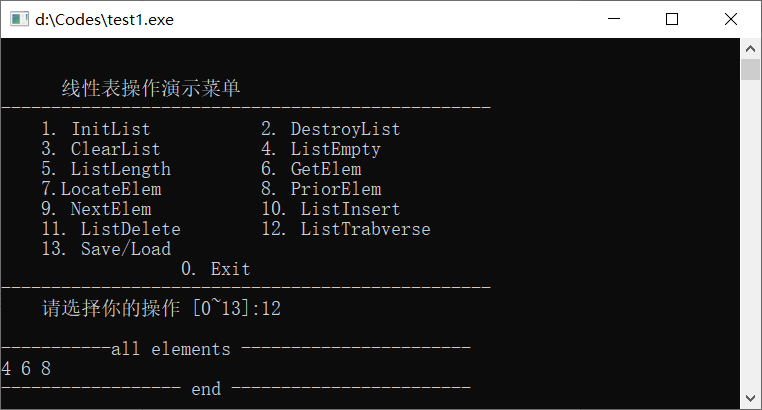


**第八步：9.获取后继**

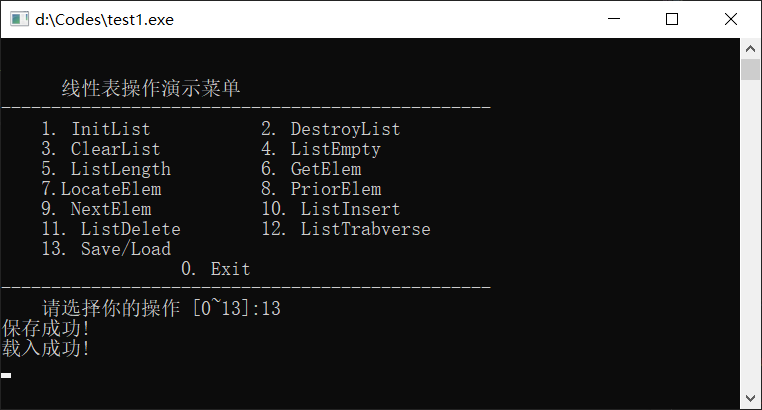




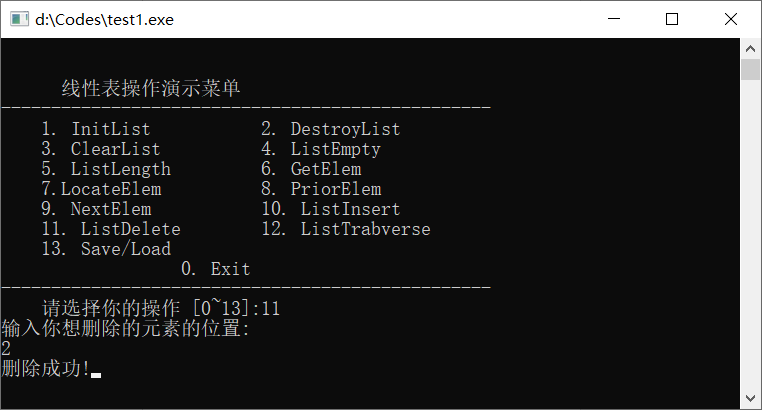
**第九步：12.遍历线性表**



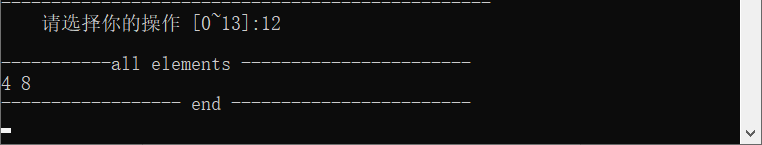
**第十步：13.文件保存/文件读取**



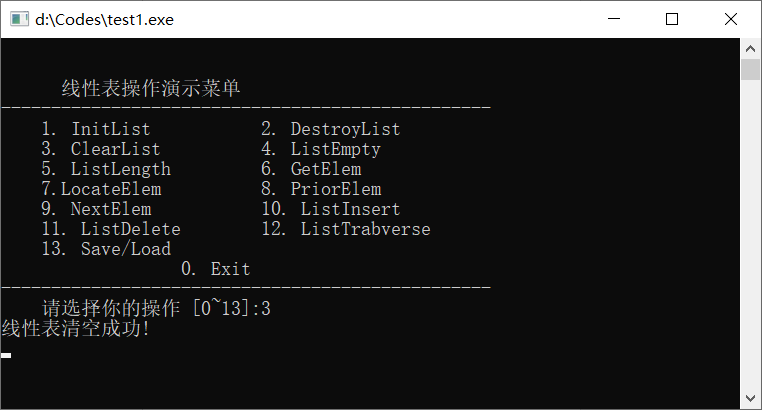
**第十一步：11.删除元素**



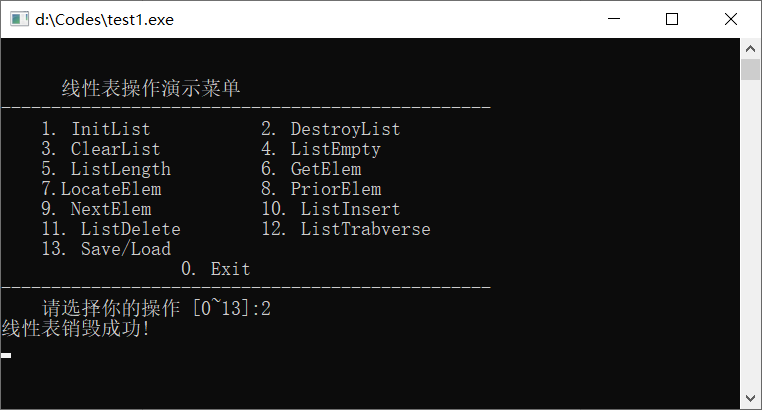
线性表现状：



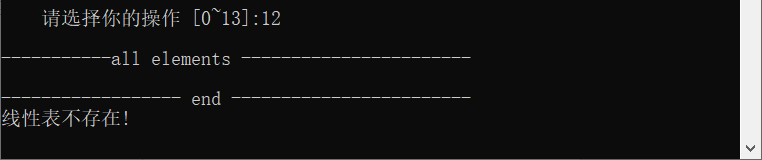
**第十二步：3.清空线性表**



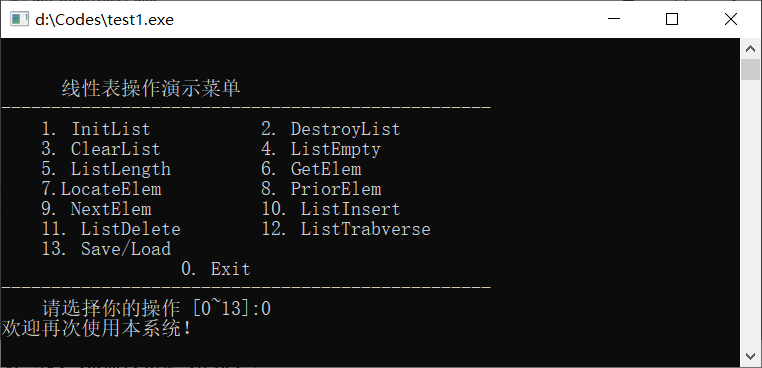
**第十三步：2.销毁线性表**



线性表现状：



**第十四步：0.退出系统**



**1.2.2 多线性表操作**

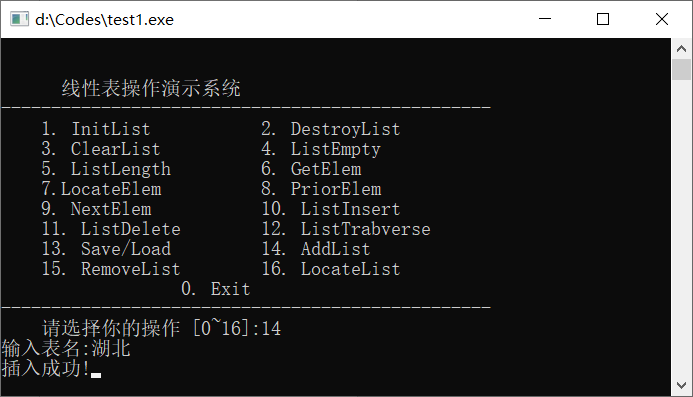
初始界面选择“b：多线性表”后显示如下菜单

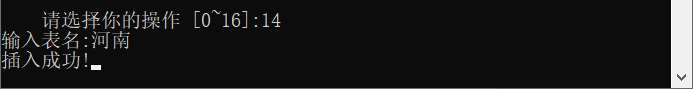


测试内容如下表所示

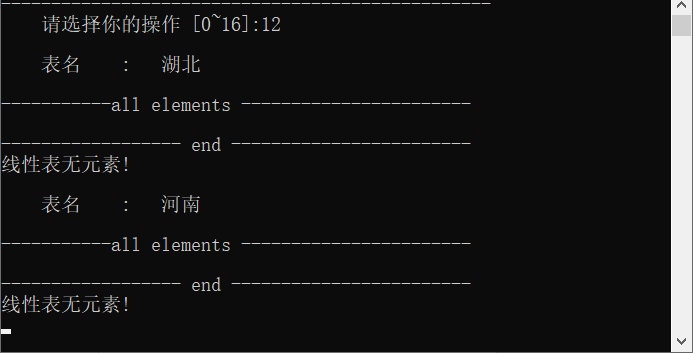
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试功能**  **及其对应序号** | **要管理的**  **线性表序号** | **输入** | **预期输出** | **当前操作完成后**  **多线性表状态** |
| 14.添加线性表  （2次） | 1和2 | 湖北  河南  （表名） | 插入成功!  （2次） | 多表内有两个空表，表名分别为湖北和河南 |
| 10.插入元素 | 1和2 | 1；1 2 3 （湖北）  2；4 5 6 （河南） | 插入成功  （6次） | 湖北：1 2 3  河南：4 5 6 |
| 16.查找线性表 | 1和2 | 河南 | 位置在2 | 同上 |
| 15.移除线性表 | 1 | 湖北 | 移除成功！ | 河南：4 5 6 |

**第一步：14.添加线性表**

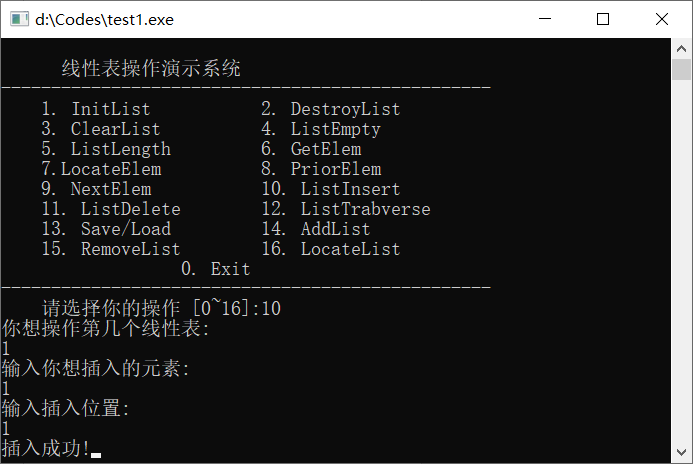
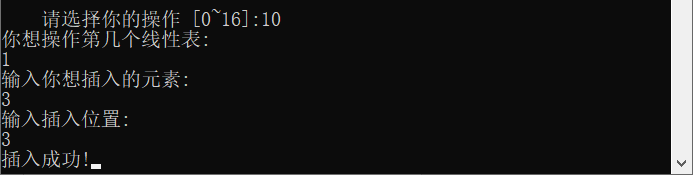
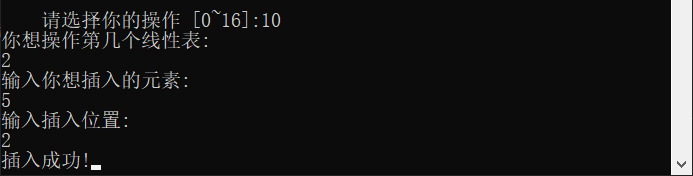




多表状态：

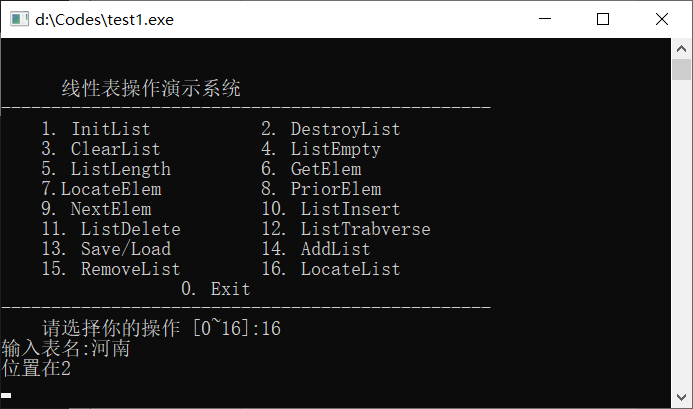


**第一步：10.插入元素**

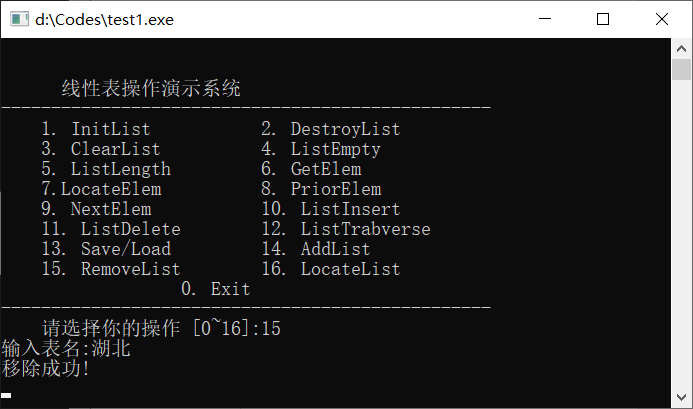
  

多表状态：

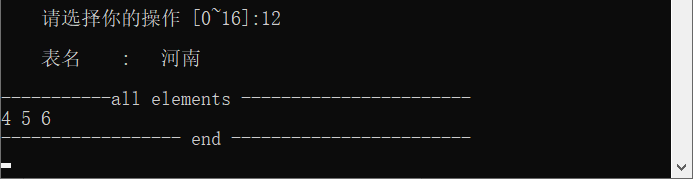
**第二步：16.查找线性表**



**第三步：15.移除线性表**



多表现状：



**附加功能测试样例：-2, 1, -3, 4, -1, 2, 1, -5, 4**

1. 最大连续子数组和结果为6
2. 和为-1的连续子数组个数为4个
3. 从小到大排序结果为-5, -3, -2, -1, 1, 1, 2, 4, 4

# 2基于链式存储结构的线性表实现

## 2.1 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物理结构采用单链表,熟练掌握线性表的基本运算的实现。

## 2.2 系统测试用例

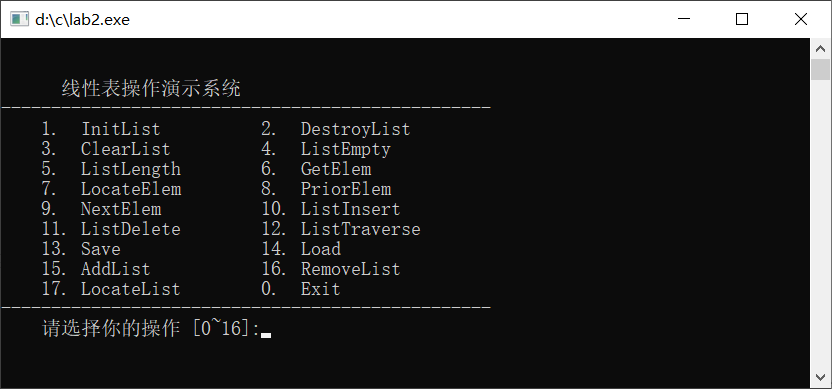


图 2-1 系统界面

**测试内容如下表所示**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能**  **及其对应序号** | **输入** | **预期输出** | **当前操作完成后**  **线性表状态** |
| 15.AddList | 表一（表名） | 插入成功! | 一个名为表一的空表 |
| 10.ListInsert | 1 6 1  1 7 2  1 8 3  （操作表序 插入元素 插入位置） | 插入成功！（3次） | 表一：6 7 8 |
| 4.ListEmpty | \ | 线性表不为空! | 同上 |
| 5.ListLength | \ | 线性表的长度是 3 | 同上 |
| 6.GetElem | 2（要获取的元素的位序） | 第2个元素是7 | 同上 |
| 7.LocateElem | 8（要定位的元素） | 该数字在第3位 | 同上 |
| 8.PriorElem | 6（要获取前驱的元素） | 这个元素没有前驱! | 同上 |
|  | 7（要获取前驱的元素） | 前驱是6 | 同上 |
| 9.NextElem | 8（要获取后继的元素） | 这个元素没有后继! | 同上 |
|  | 6（要获取后继的元素） | 后继是7 | 同上 |
| 12.ListTraverse | \ | 6 7 8 | 同上 |
| 13.Save | \ | 保存成功! | 同上（将线性表中的数据保存到test.txt中） |
| 14.Load | 读取 | 载入成功！ | 表一：6 7 8  读取：6 7 8 |
| 11. ListDelete | 2（删除第二个表中的元素）  2（要删除元素的位置） | 删除成功！ | 表一：6 7 8  读取：6 8 |
| 3. ClearList | 1 | 线性表清空成功! | 表一：\  读取：6 8 |
| 16. RemoveList | 1 | 移除成功！ | 读取：6 8 |
| 15.AddList | 新表  （插入表的表名） | 插入成功！ | 读取：6 8  新表：\ |
| 17. LocateList | 读取（要定位的表的表名） | 位置在1 | 同上 |

**第一步：15.插入一个单线性表**

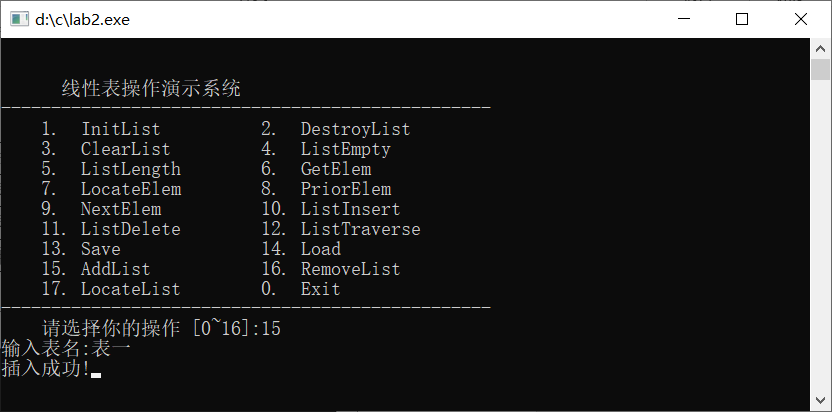


图 2-2 插入线性表

**第二步：10.向“表一”中插入元素（3次）**

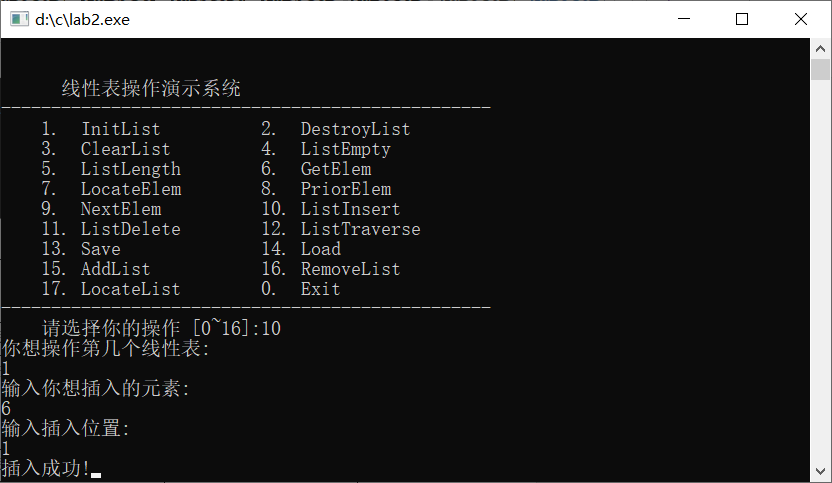


图 2-3 插入元素

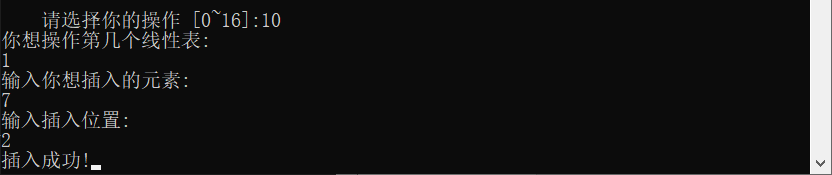


图 2-4 插入元素

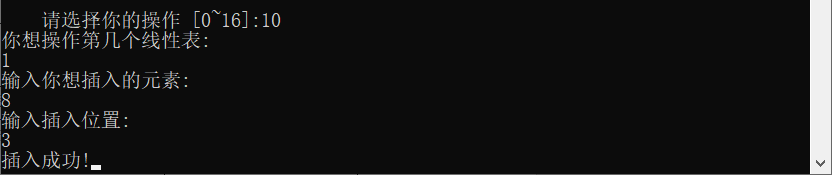


图 2-5 插入元素

**第三步：4.判空线性表表一**

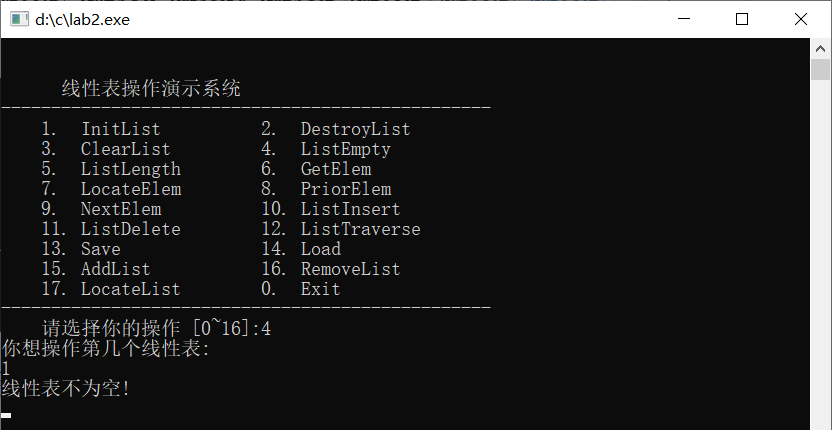


图 2-6 判空线性表

**第四步：5.求表一的表长**

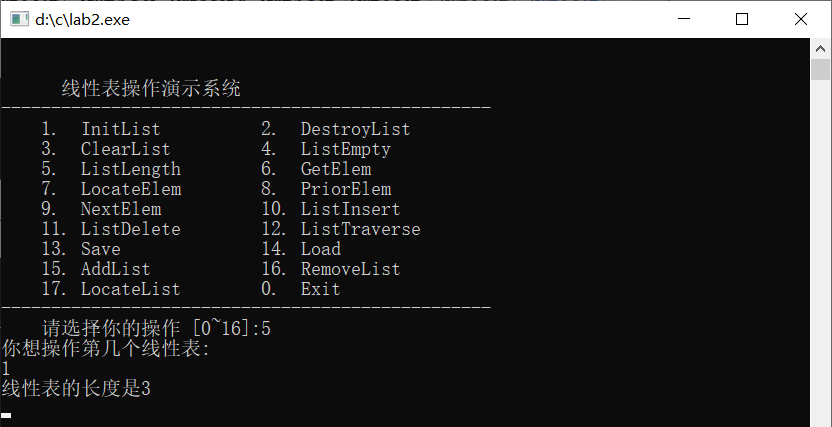


图 2-7 求表长

**第五步：6.获取表一中位序为2的元素**

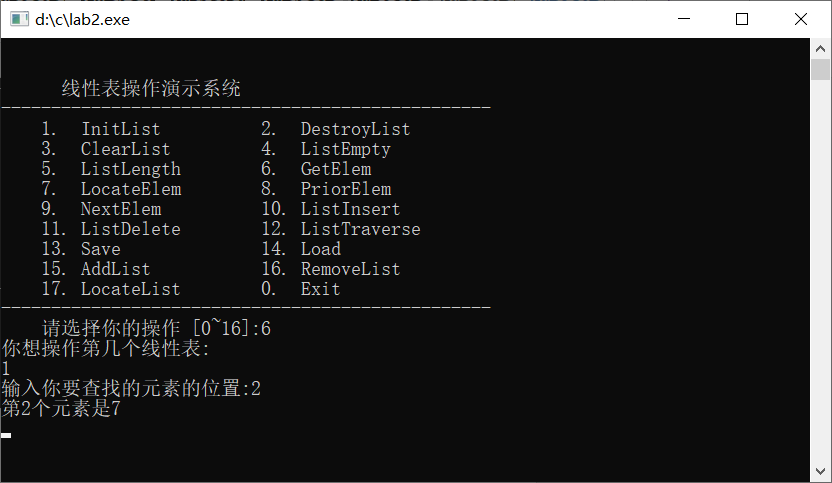


图 2-8 获取元素

**第六步：7.定位表一中元素8的位序**

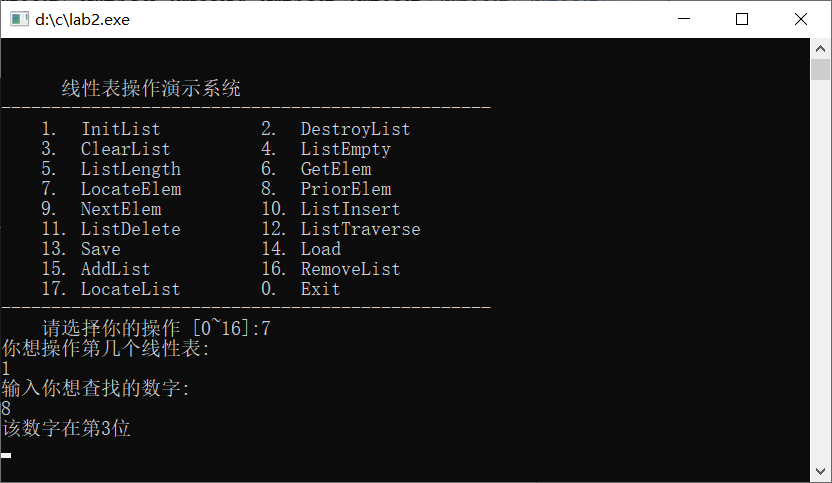


图 2-9 定位元素

**第七步：8.获取第一个元素6和第二个元素7的前驱**

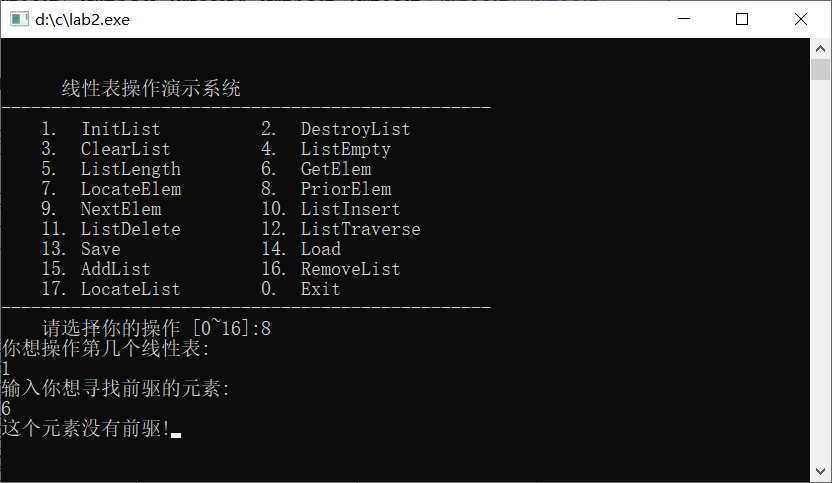


图 2-10 获取前驱

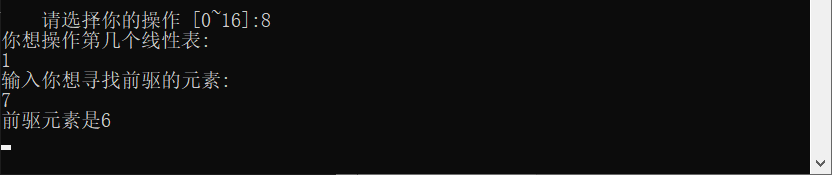


图 2-11 获取前驱

**第八步：9.获取第一个元素6和第三个元素8的后继**

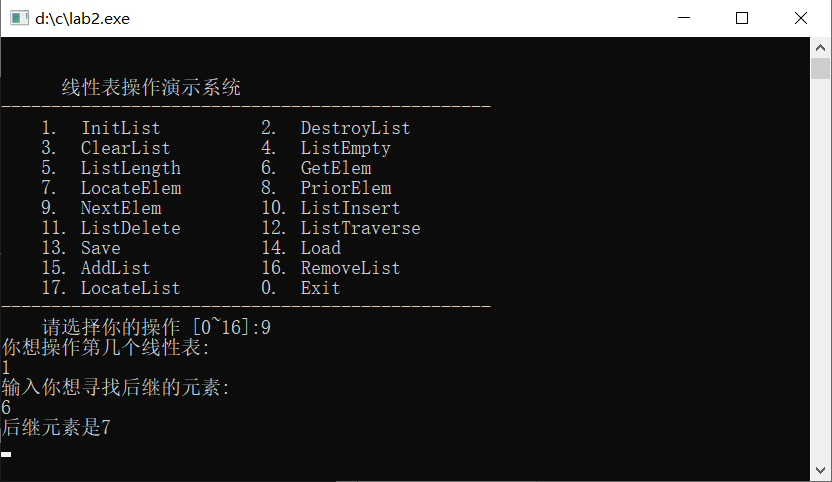


图 2-12 获取后继

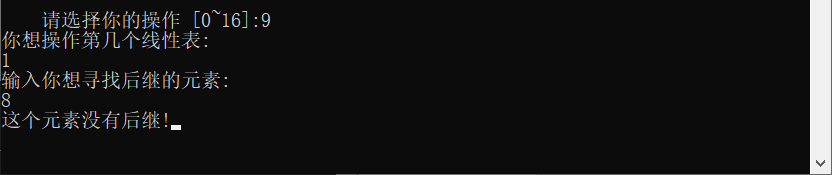


图 2-13 获取后继

**第九步：12.遍历线性表（即表一）**

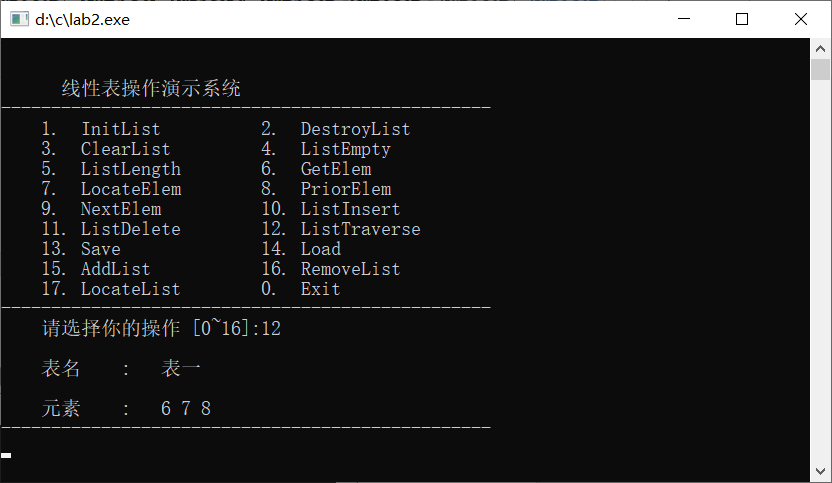
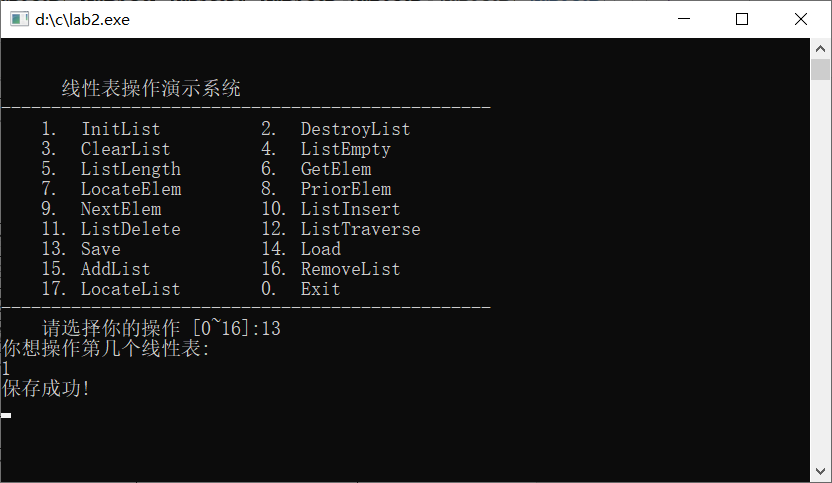


图 2-14 遍历线性表

**第十步：13.将表一保存到test.txt中以便读取**

图 2-15 文件保存

**第十一步：14.文件读取 将新表命名为“读取”**

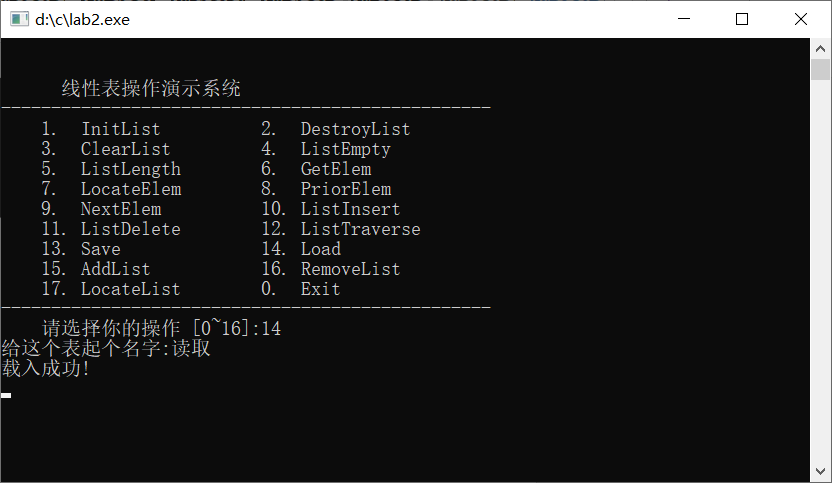


图 2-16 文件读取

线性表现状：

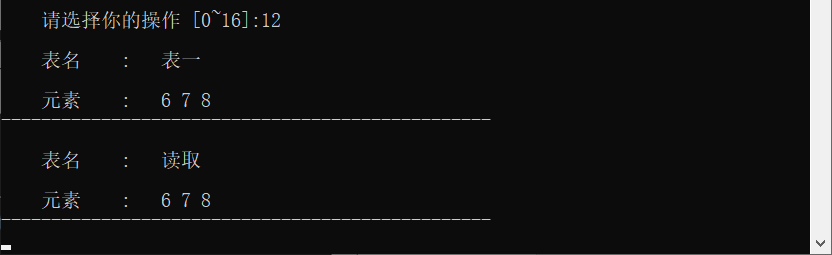
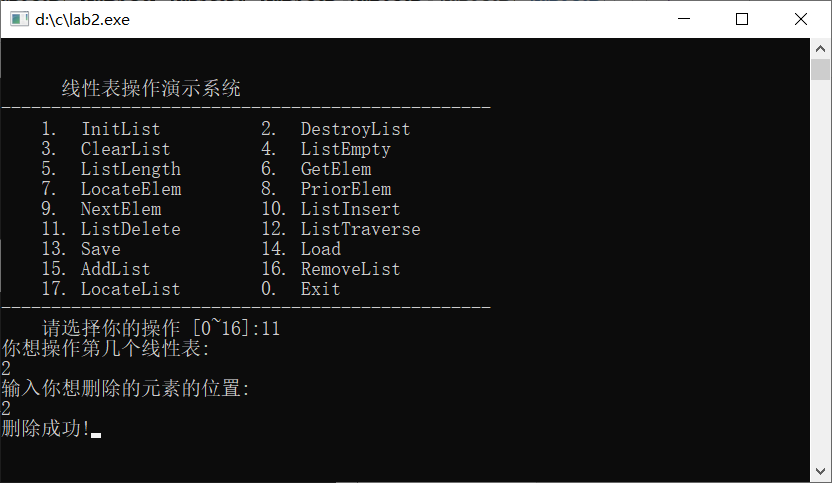


图 2-17 现状

**第十二步：11.删除第二个表“读取”的第二个元素7**

图 2-18 删除元素

表的现状：

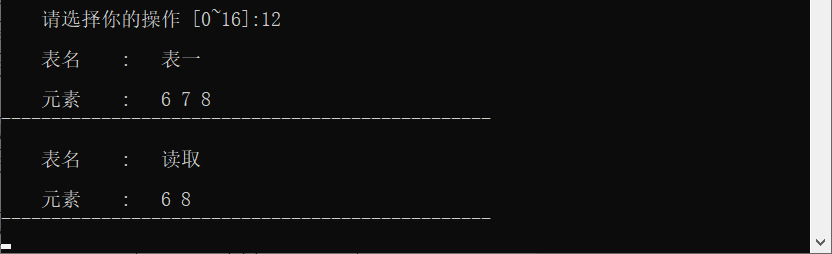


图 2-19 现状

**第十三步：3.清空第一个线性表“表一”**

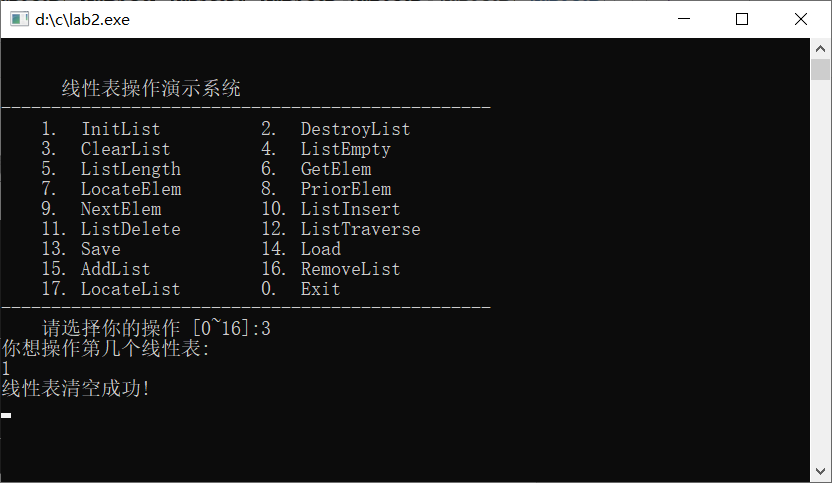
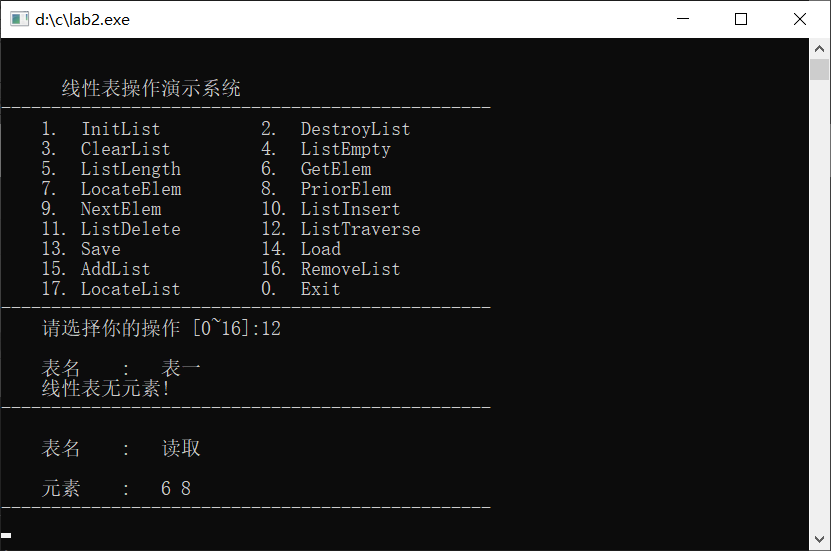


图 2-20 清空线性表

表的现状：

图 2-21 现状

**第十四步：16.移除第一个线性表“表一”**

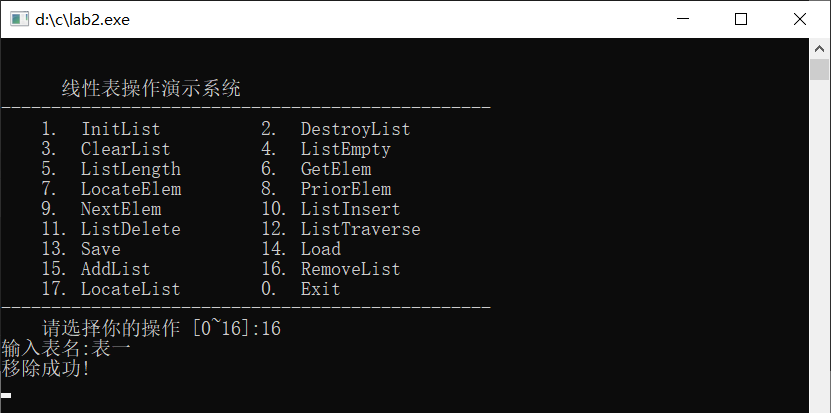
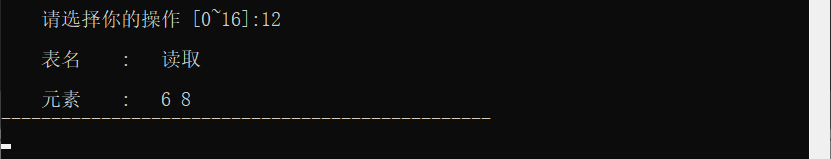
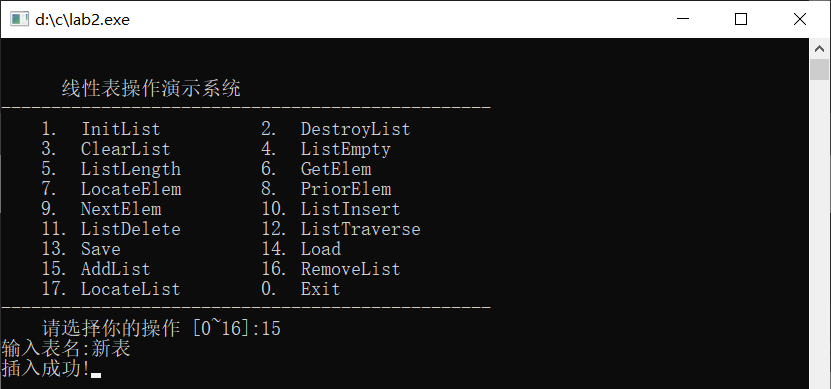


图 2-22 移除线性表

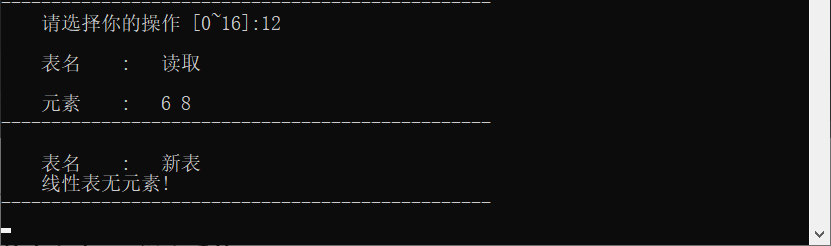
表的现状：

图 2-23 现状

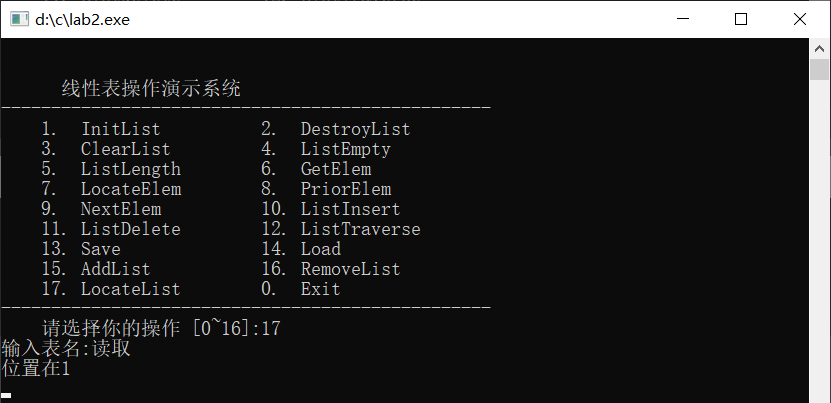
**第十五步：15.添加一个新表名为“新表”**

图 2-24 添加一个“新表”

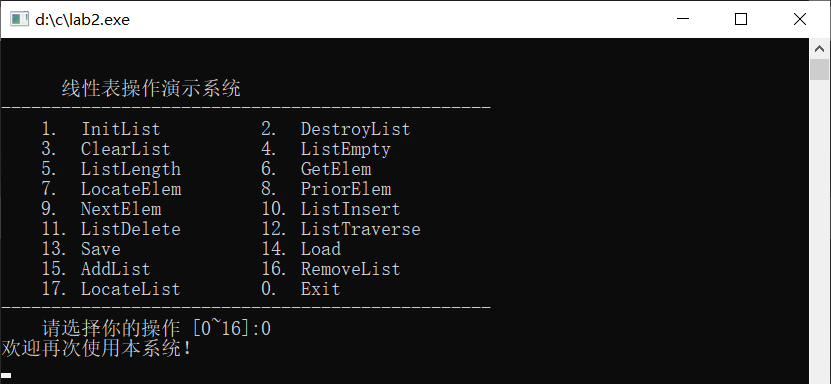
表的现状：

图 2-25 现状

**第十六步：17.定位名为“读取”的线性表的位序**

图 2-26 定位线性表

**第十七步：0.退出系统**

图 2-27 退出系统

**附加功能测试样例：-2, 1, -3, 4, -1, 2, 1, -5, 4**

**1、**链表反转结果为4，-5，1，2，-1，4，-3，1，-2

2、删除链表倒数第2个节点结果为4，-5，1，2，-1，4，-3，-2

3、链表从小到大排序结果为-5, -3, -2, -1, 1, 1, 2, 4, 4

# 3基于二叉链表的二叉树实现

## 3.1 实验目的

通过实验达到⑴加深对二叉树的概念、基本运算的理解；⑵熟练掌握二叉树的逻辑结构与物理结构的关系；⑶以二叉链表作为物理结构，熟练掌握二叉树基本运算的实现。

## 3.2 系统测试用例

本实验将单二叉树操作和多二叉树管理操作合并，单二叉树操作可通过对多二叉树中的某个单二叉树进行相应操作来实现。

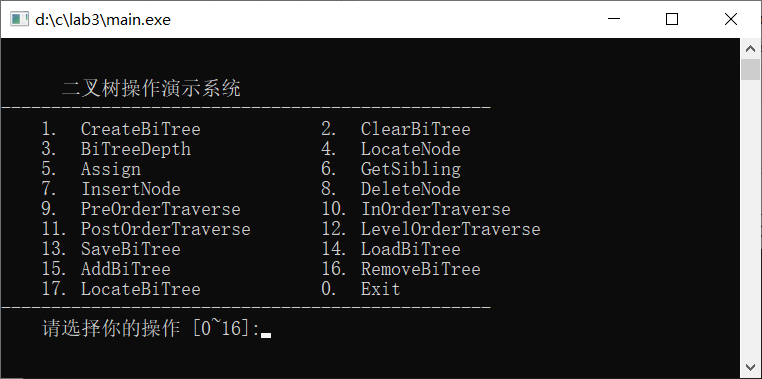


图 3-1 系统界面

**测试内容如下表所示**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能**  **及其对应序号** | **输入** | **预期输出** | **当前操作完成后**  **多二叉树状态** |
| 15. AddBiTree | “树a”（树名）  “1 1 a 2 2 b 3 3 c 6 4 d 7 5 e 0 0 null”（数据） | 插入成功！ | 见下图3-2-2 |
| 7. InsertNode | “1 6 f 3 0” | 插入操作成功 | 见下图3-2-4 |
| 3. BiTreeDepth | “1” | 树的深度是4 | 同上 |
| 4. LocateNode | “1 6” | 该关键字信息为：6,f | 同上 |
| 5. Assign | “1 5 11 x” | 赋值成功! | 见下图3-2-8 |
| 6. GetSibling | “1 3” | 兄弟节点为：2,b | 同上 |
| 8. DeleteNode | “1 2” | 删除操作成功 | 见下图3-2-11 |
| 9.PreOrderTraverse | \ |  | 同上 |
| 10.InOrderTraverse | \ |  | 同上 |
| 11.PostOrderTraverse | \ |  | 同上 |
| 12.LevelOrderTraverse | \ |  | 同上 |
| 13. SaveBiTree | “1” | 保存成功! | 同上 |
| 14. LoadBiTree | “1 新树” | 载入成功! | 见下图3-2-14 |
| 16. RemoveBiTree | “树a” | 移除成功! | 见下图3-2-17 |
| 17. LocateBiTree | “新树” | 位置在1 | 同上 |

**第一步：15.插入一个单二叉树**

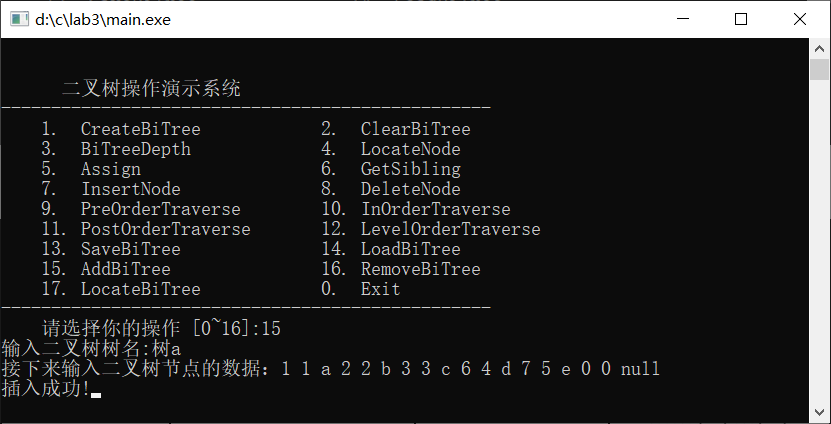


图 3-2 插入二叉树

**现状：**

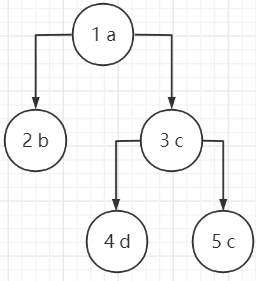


图 3-3

**第二步：10.向“树a”中插入节点**



图 3-4 插入节点

**现状：**

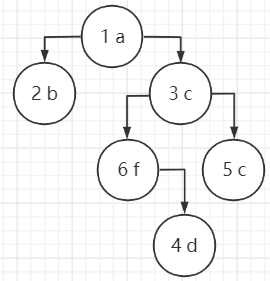


图 3-5

**第三步：3.求“树a”的深度**

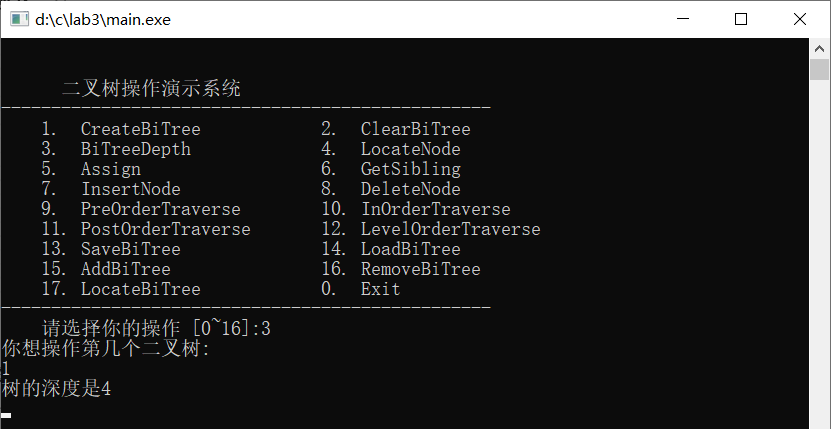


图 3-6 求二叉树深度

**第四步：4.查找关键字为6的节点**

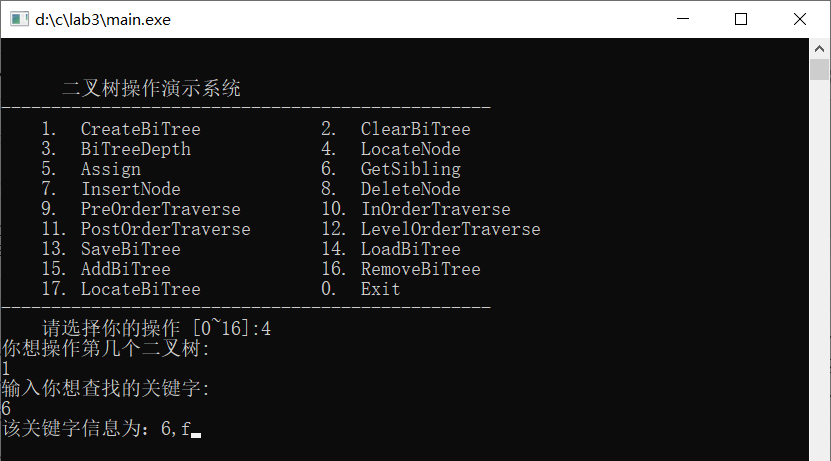


图 3-7 查找节点

**第五步：5.将原本关键字为5的节点赋值为11**



图 3-8 节点赋值

**现状：**

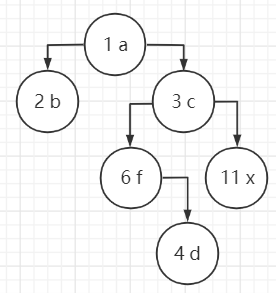


图 3-9

**第六步：6.获取“3 c”的兄弟节点**



图 3-10 获取兄弟节点

**第七步：8.删除关键字为2的节点**

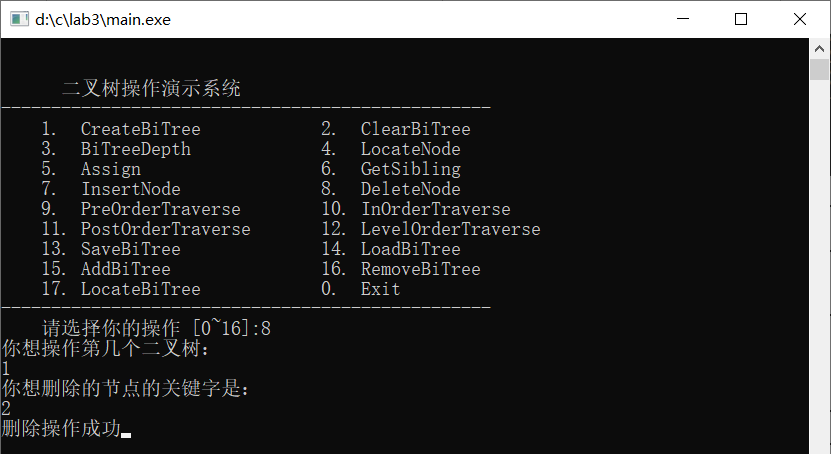


图 3-11 删除节点

**现状：**

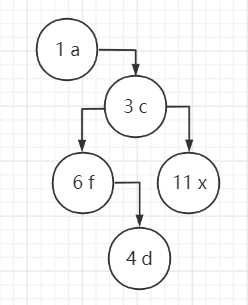


图 3-12

**第八到十一步：前中后层序遍历二叉树**

测试图见上表格

**第十二步：将“树a”保存在文件中**

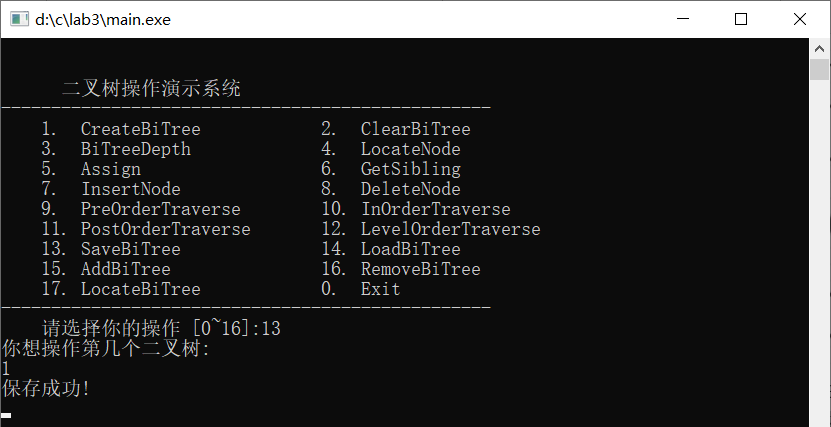


图 3-13 文件保存

**第十三步：从文件中读取二叉树并命名为“新树”**

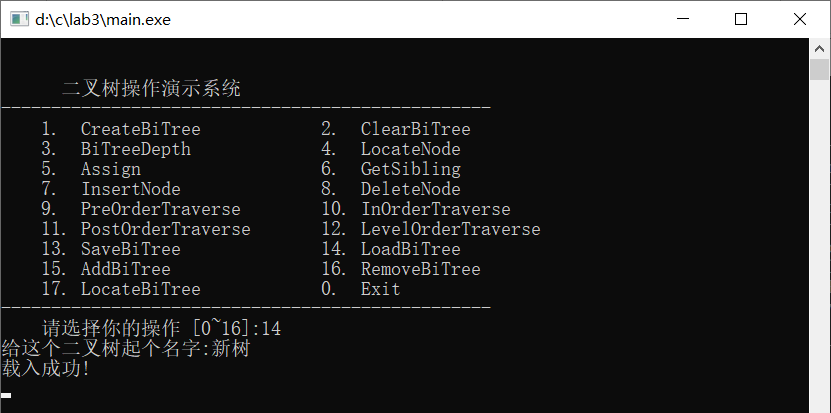


图 3-14 文件读取

**现状：**

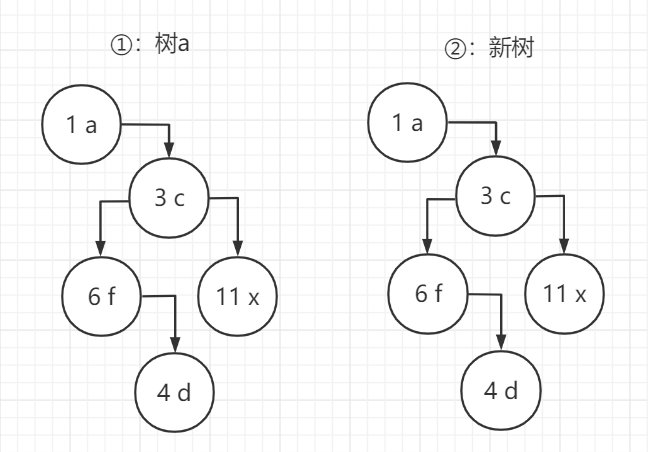


图 3-15

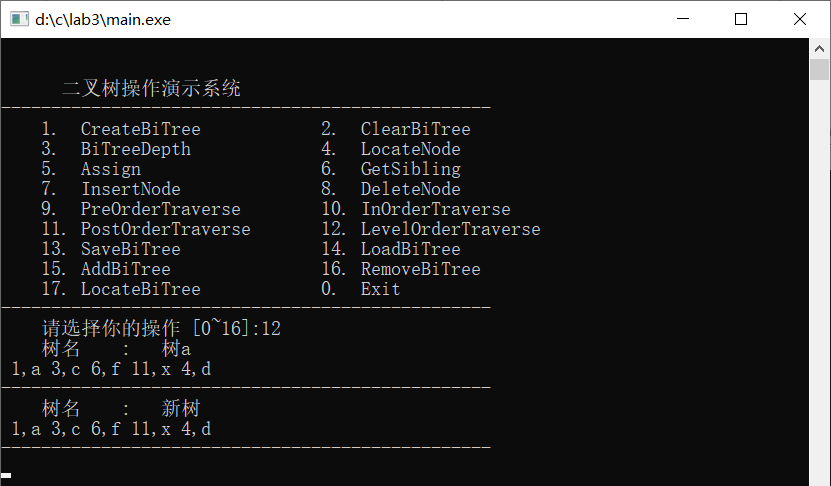


图 3-16 此刻的层序遍历

**第十四步：移除第一个二叉树“树a”**

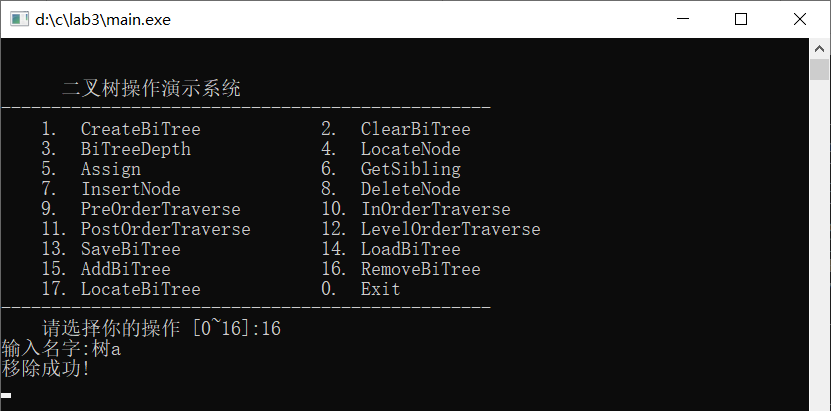


图 3-17 移除“树a”

**现状：**

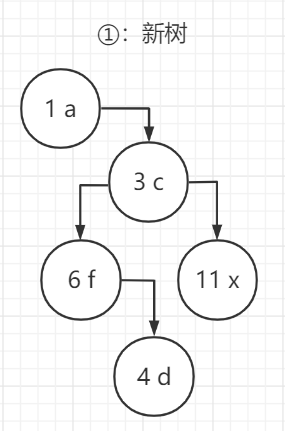


图 3-18

**第十五步：查找名为“新树”的二叉树的位序**

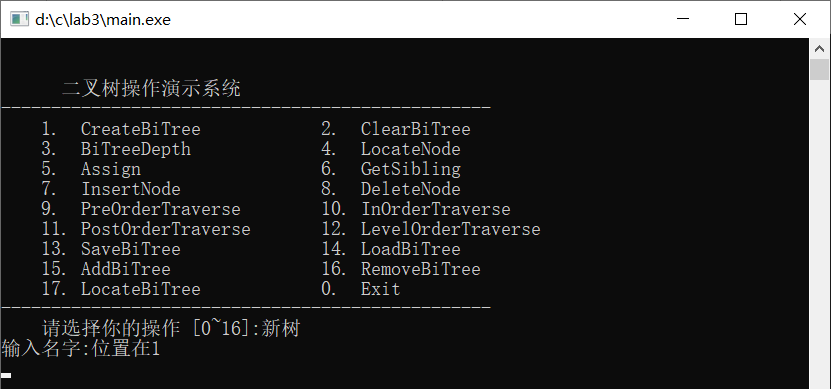


图 3-19 查找二叉树

**使用上述二叉树，附加功能结果为：**

**1、**最大路径和为14

2、d，x的最近公共祖先为c

3、翻转二叉树的前序遍历结果为：1 a，3 c，11 x，6 f，4 d

# 4基于邻接表的图实现

## 4.1 实验目的

通过实验达到⑴加深对图的概念、基本运算的理解；⑵熟练掌握图的逻辑结构与物理结构的关系；⑶以邻接表作为物理结构，熟练掌握图基本运算的实现。

## 4.2 系统测试用例

本实验将单无向图操作和多图管理操作合并，单无向图操作可通过对多图中的某个单无向图进行相应操作来实现。

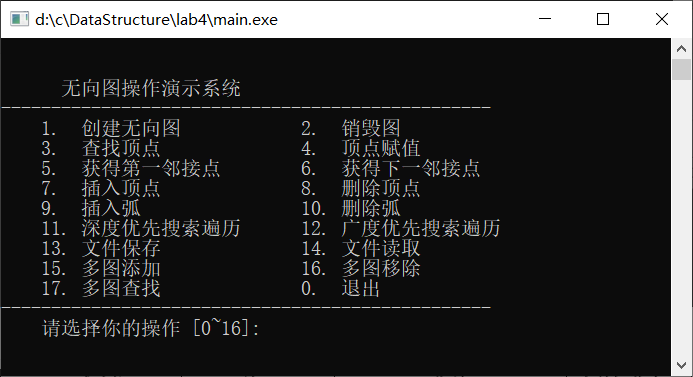


图 4-1 系统界面

**测试内容如下表所示**

|  |  |  |  |
| --- | --- | --- | --- |
| **测试功能**  **及其对应序号** | **输入** | **预期输出** | **当前操作完成后**  **多二叉树状态** |
| 15. 多图添加 | “图1；5 线性表 8 集合 7 二叉树 6 无向图 -1 nil 5 6 5 7 6 7 7 8 -1 -1” | 添加成功！ | 见下图4-2 |
| 3. 查找顶点 | “8” | 该顶点信息：8 集合 | 见下图4-4 |
| 5. 获得第一邻接点 | “7” | 第一邻接点信息：8 集合 | 见下图4-5 |
| 6. 获得下一邻接点 | “7 8” | 下一邻接点信息：6 无向图 | 见下图4-6 |
| 4. 顶点赋值 | “6 9 有向图” | 赋值成功! | 见下图4-7 |
| 8. 删除顶点 | “9” | 删除操作成功 | 见下图4-9 |
| 7. 插入顶点 | “6 无向图” | 插入成功 | 见下图4-11 |
| 11.DFS | / |  | 见下图4-13 |
| 12.BFS | / |  | 见下图4-14 |
| 13. 文件保存 | / | 保存成功 | 见下图4-15 |
| 14. 文件读取 | / | 载入成功 | 见下图4-16 |
| 17. 多图查找 | “新图” | 位置在2 | 见下图4-18 |
| 16. 多图移除 | “图1” | 移除成功 | 见下图4-19 |

**第一步：15.插入一个单无向图**

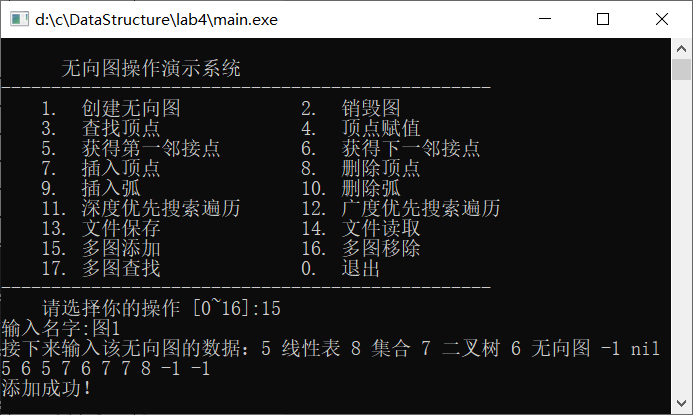


图 4-2 第一步

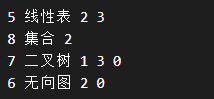


图 4-3 现状

**第二步：3.查找关键字为8的顶点**



图 4-4 第二步

**第三步：5.查找7的下一邻接点**



图 4-5 第三步

**第四步：6.查找7的表结点8下一邻接点**

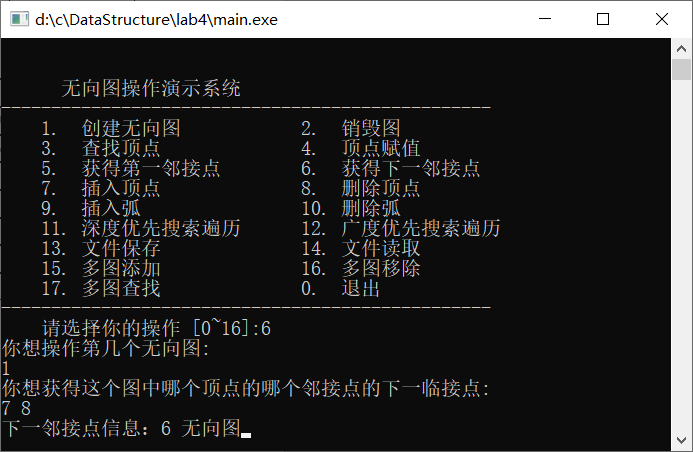


图 4-6 第四步

**第五步：4.将关键字为6的顶点赋值为“9 无向图”**



图 4-7 第五步



图 4-8 结点现状

**第六步：8.删除关键字为9的节点**

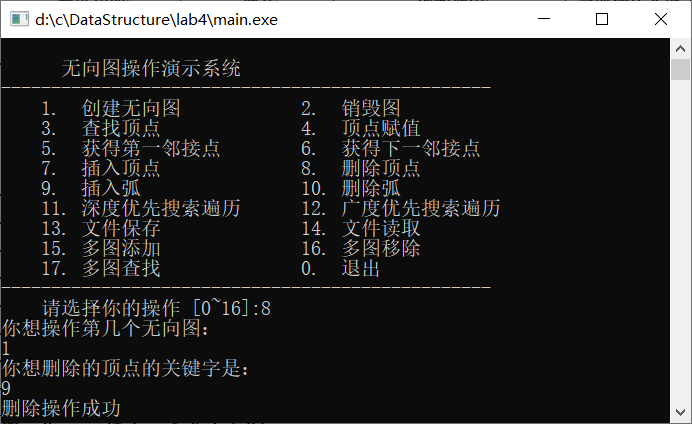


图 4-9 第六步



图 4-10 BFS现状

**第七步：7.插入信息为“6 无向图”的顶点**

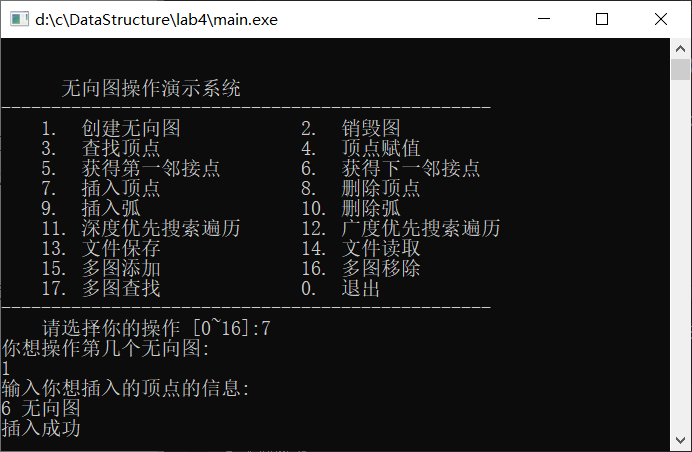


图 4-11 第七步



图 4-12 BFS现状

**第八步：11.DFS**



图 4-13 第八步

**第九步：12.BFS**



图 4-14 第九步

**第十步：13.将“图1”保存在本地文件中**

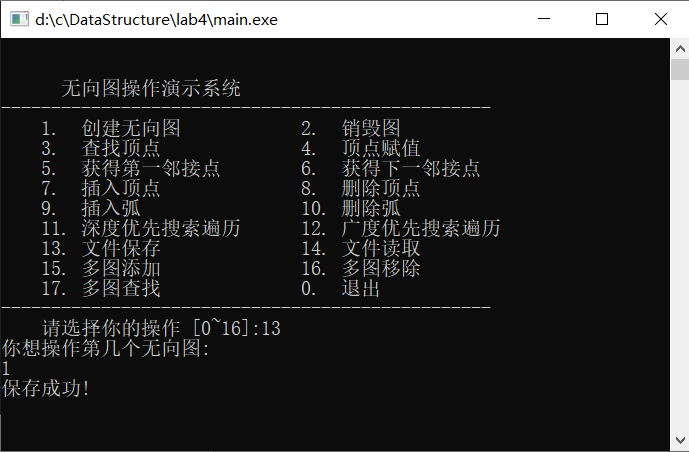


图 4-15 第十步

**第十一步：14.读取本地文件创建新图并命名为：新图**

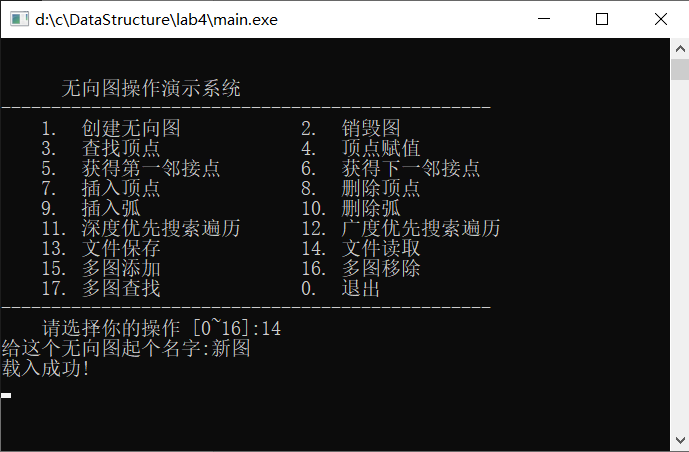


图 4-16 第十一步

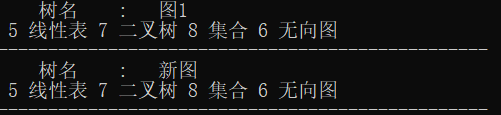


图 4-17 BFS现状

**第十二步：17.查找名为“新图”的无向图**

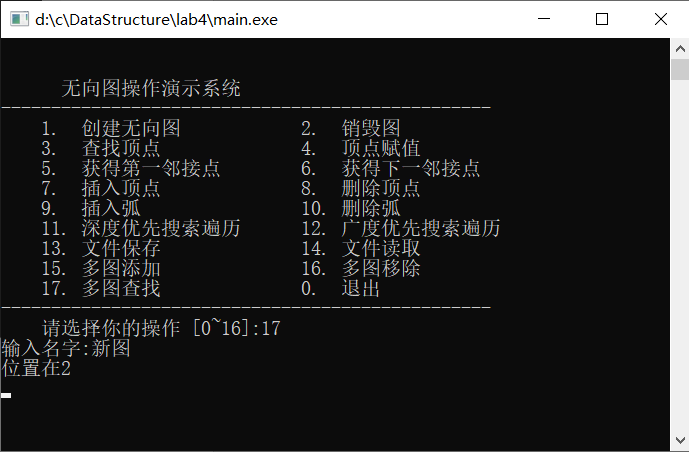


图 4-18 第十二步

**第十三步：16.删除名为“图1”的无向图**

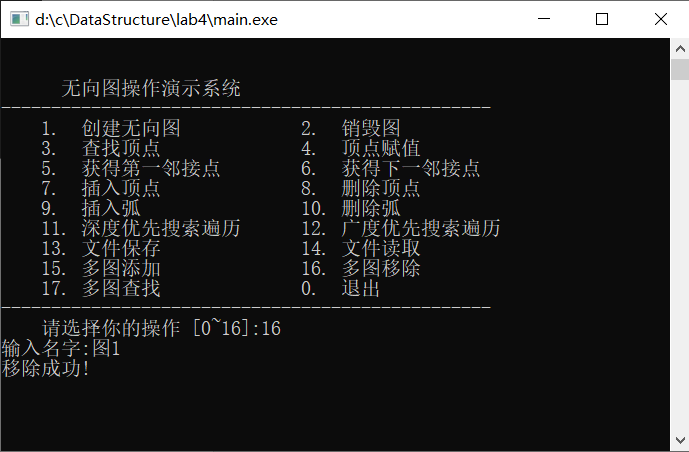


图 4-19 移除图1



图 4-20 BFS现状

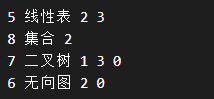


图4-21 图邻接表现状

**使用如图4-21所示图，附加功能结果为：**

1. 距离小于k的顶点集合：函数名称是VerticesSetLessThanK(G,v,k)，初始条件是图G存在；操作结果是返回与顶点v距离小于k的顶点集合；

输入：图名，集合，2

输出：二叉树

1. 顶点间最短路径和长度：函数名称是ShortestPathLength(G,v,w); 初始条件是图G存在；操作结果是返回顶点v与顶点w的最短路径的长度；

输入：图名，集合，线性表

输出：2

1. 图的连通分量：函数名称是ConnectedComponentsNums(G)，初始条件是图G存在；操作结果是返回图G的所有连通分量的个数；

输入：图名

输出：1

将集合与二叉树之间的边断开后

输入：图名

输出：2