

**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202003**

**学 号： U202015375**

**姓 名： 汪宇飞**

**指导教师： 陈奇**

**报告日期： 202X年 X月 X 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 3](#_Toc458159879)

[1.1 问题描述 3](#_Toc458159880)

[1.2 系统设计 3](#_Toc458159882)

[1.3 系统实现 5](#_Toc458159883)

[1.4 系统测试 8](#_Toc458159883)

[1.5 实验小结 29](#_Toc458159884)

[2 基于链式存储结构的线性表实现 2](#_Toc458159885)

[2.1 问题描述 2](#_Toc458159886)

[2.2 系统设计 2](#_Toc458159887)

[2.3 系统实现 2](#_Toc458159888)

[2.4 实验小结 2](#_Toc458159889)

[3 基于二叉链表的二叉树实现 2](#_Toc458159890)

[3.1 问题描述 2](#_Toc458159891)

[3.2 系统设计 2](#_Toc458159892)

[3.3 系统实现 2](#_Toc458159893)

[3.4 实验小结 2](#_Toc458159894)

[4 基于二叉链表的二叉树实现 2](#_Toc458159895)

[4.1 问题描述 2](#_Toc458159896)

[4.2 系统设计 2](#_Toc458159897)

[4.3 系统实现 2](#_Toc458159898)

[4.4 实验小结 2](#_Toc458159899)

[参考文献 2](#_Toc458159900)

[附录A 基于顺序存储结构线性表实现的源程序 2](#_Toc458159901)

[附录B 基于链式存储结构线性表实现的源程序 2](#_Toc458159902)

[附录C 基于二叉链表二叉树实现的源程序 2](#_Toc458159903)

[附录D 基于邻接表图实现的源程序 2](#_Toc458159904)

# 1 基于顺序存储结构的线性表实现

## 1.1 问题描述

本次实验所要解决的内容是采用顺序表作为线性表的物理结构，实现线性表抽象数据类型（即ADT）的基本运算，构造一个具有菜单的功能演示系统，并要求实现线性表的文件形式保存和线性表管理。

### 顺序表ADT实现线性表

根据实验要求，以顺序存储结构为基础，构造函数实现以下功能：线性表的创建、销毁线性表、清空线性表、线性表判空、线性表长度、获取元素、查找元素、获取前驱元素、获取后继元素、插入元素、删除元素、遍历线性表、线性表读写文件。

### 构造具有菜单的功能演示系统

根据实验要求，编写具有菜单的演示系统，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，并给出适当的操作提示显示。

### 实现线性表的文件形式保存

根据实验要求，构造将线性表写入文件和线性表读取文件的函数，并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。

### 实现多个线性表的管理

根据实验要求，为实现多个线性表的管理，设计线性表集合的结构，并编写包括新增线性表、删除线性表、查找线性表等函数加以管理。

## 1.2 系统设计

根据实验要求，整体系统结构如图1-1所示：

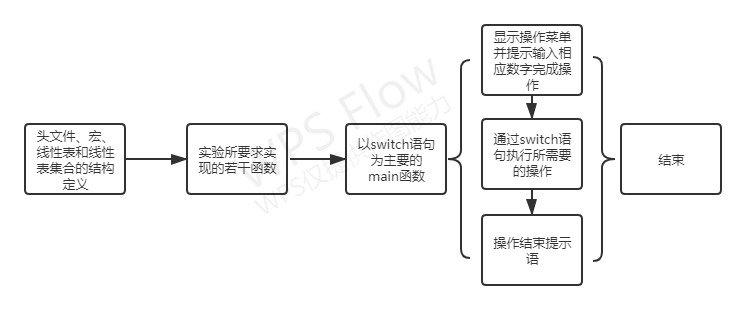


图1-1 整体系统结构设计

数据结构设计如图1-2、图1-3所示：

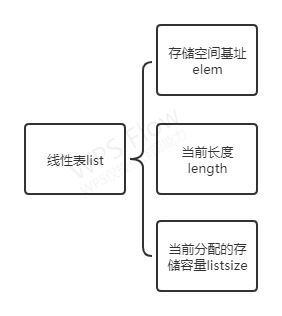


图1-2 线性表list的结构设计

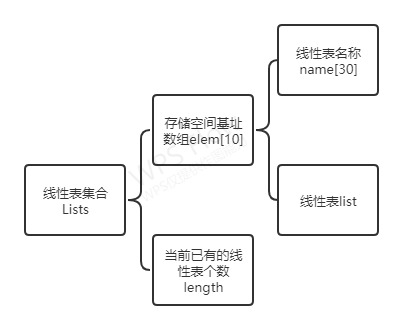


图1-3 线性表集合Lists的结构设计

## 1.3 系统实现

表1-1 主要函数目录

|  |  |  |  |
| --- | --- | --- | --- |
| 选项 | 功能操作 | 选项 | 功能操作 |
| 1 | 创建线性表 | 2 | 销毁线性表 |
| 3 | 清空线性表 | 4 | 线性表判空 |
| 5 | 线性表长度 | 6 | 获取元素 |
| 7 | 查找元素 | 8 | 获取前驱元素 |
| 9 | 获取后继元素 | 10 | 插入元素 |
| 11 | 删除元素 | 12 | 遍历线性表 |
| 13 | 线性表写入文件 | 14 | 线性表读取文件 |
| 15 | 增加一个线性表 | 16 | 删除一个线性表 |
| 17 | 查找线性表 | 0 | 退出 |

1. 创建线性表：首先判断线性表是否存在，若存在则返回INFEASIBLE。否则，使用malloc函数给elem分配基址，length置为0，listsize置为LIST\_INIT\_SIZE,最终返回OK。
2. 销毁线性表：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，length和listsize均置为0，对elem执行free操作，并使elem为NULL，最终返回OK。
3. 清空线性表：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则进入for循环，将每个元素置为0，最终返回OK。
4. 线性表判空：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，判断length是否为0，是则返回TRUE，否则返回FALSE。
5. 线性表长度：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，返回length。
6. 获取元素：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，判断i的取值范围是否合法，若不合法则返回ERROR。否则取第i个元素将其赋值给e，返回OK。
7. 查找元素：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，定义变量sta=ERROR并进入for循环，如果找到了指定元素，则令sta=i并且break，最终返回sta。
8. 获取前驱元素：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，定义变量sta=ERROR并进入for循环，如果找到了指定元素，则将其前一个元素赋值给pre、令sta=OK并break，最终返回sta。
9. 获取后继元素：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，定义变量sta=ERROR并进入for循环，如果找到了指定元素，则将其后一个元素赋值给next、令sta=OK并break，最终返回sta。
10. 插入元素：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，判断插入位置i是否合法，若不合法则返回ERROR。否则，判断线性表是否已满，若已满则定义新基址newbase，使用realloc函数分配空间，使elem置为newbase，listsize增加LISTINCREMENT。否则进入for循环，使插入位置后续元素均往后移一位，使插入元素置于指定插入位置，并使length加一，最终返回OK。



图1-4 线性表插入元素流程图

1. 删除元素：首先判断线性表是否存在，若不存在则返回INFEASIBLE。否则，判断删除位置i是否合法，若不合法则返回ERROR。否则进入for循环，使删除位置后续元素均往前移一位，并使length减一，最终返回OK。
2. 遍历线性表：进入for循环，逐个输出各个元素，最终放回length。
3. 线性表写入文件：首先判断线性表是否存在，若不存在则返回INFEASIBLE，否则以读写方式打开文件并使用fwrite函数，关闭文件，最终返回OK。
4. 线性表读取文件：首先判断线性表是否存在，若已存在则返回INFEASIBLE。否则，将length置为0，然后进入while循环并使用fread函数，每次循环使length加一，最终返回OK。
5. 增加一个线性表：首先使用strcpy函数将ListName赋值给新线性表，然后使用malloc函数为新线性表分配空间，再将线性表的length置为0、将listsize置为LIST\_INIT\_SIZE，并使线性表集合的length加一，最终返回OK。
6. 删除一个线性表：进入for循环，使用strcmp函数将线性表集合中每个线性表的名称和指定名称比较，若相同则使后续线性表均往前移一位，并使线性表集合length减一，返回OK。若for循环结束了仍未返回，则返回ERROR。
7. 查找线性表：进入for循环，使用strcmp函数将线性表集合中每个线性表的名称和指定名称比较，若相同则返回该线性表的逻辑序号。若for循环结束了仍未返回，则返回0。

**1.4 系统测试**

1. 创建线性表

用例：无

测试结果：线性表创建成功（如图1-5）

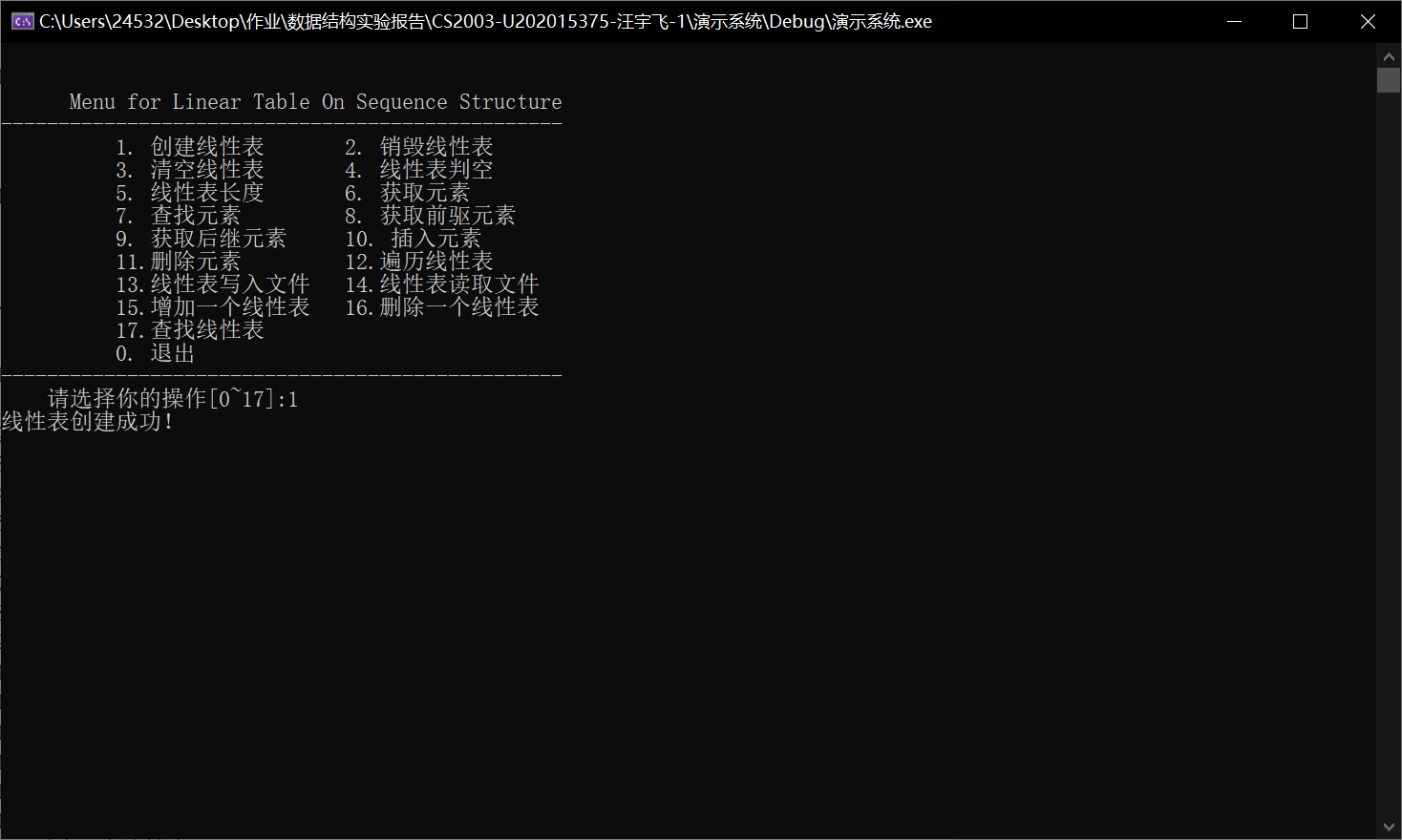


图1-5

用例：已存在线性表

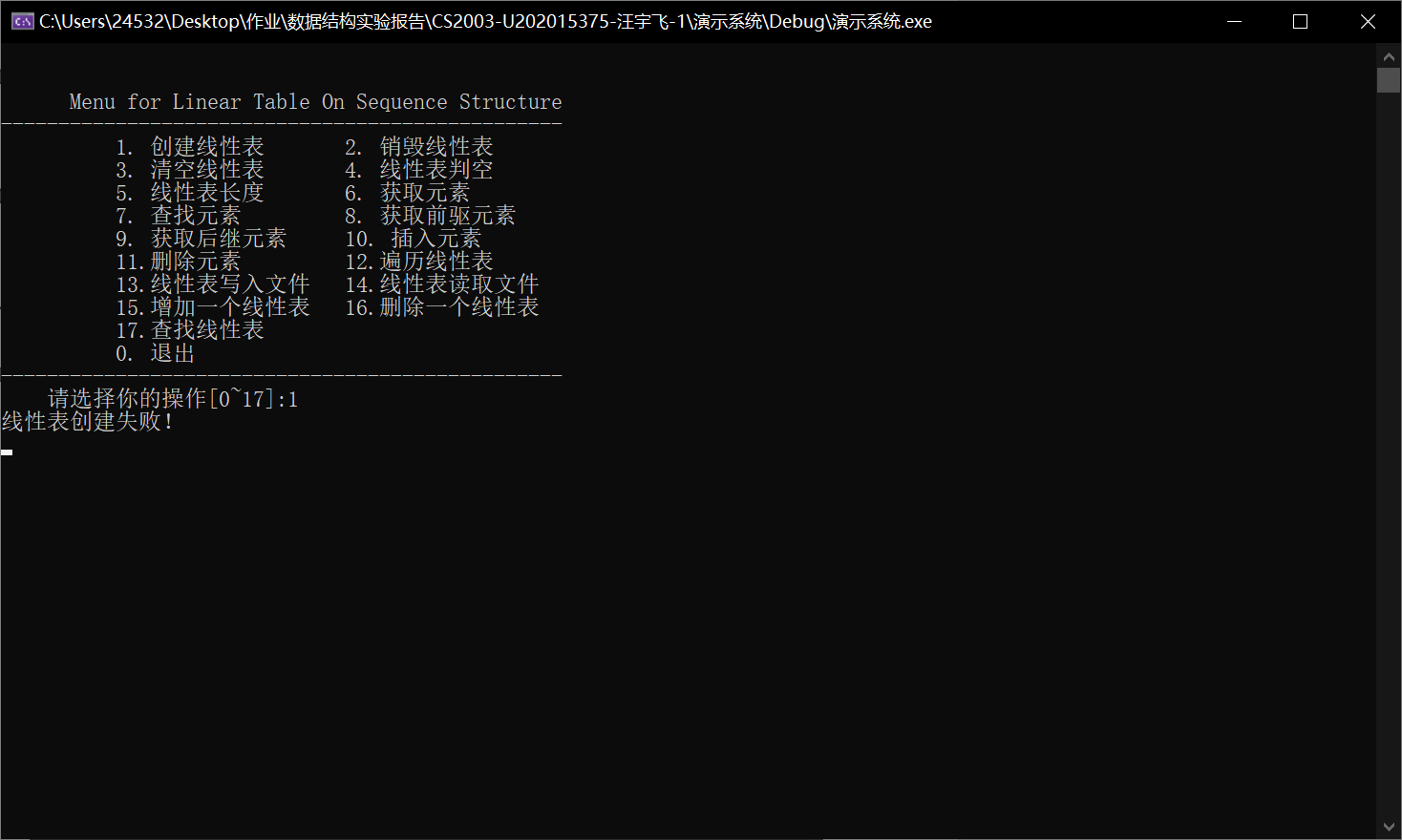
测试结果：线性表创建失败（如图1-6）

图1-6

1. 销毁线性表

用例：已存在线性表

测试结果：线性表销毁成功（如图1-7）

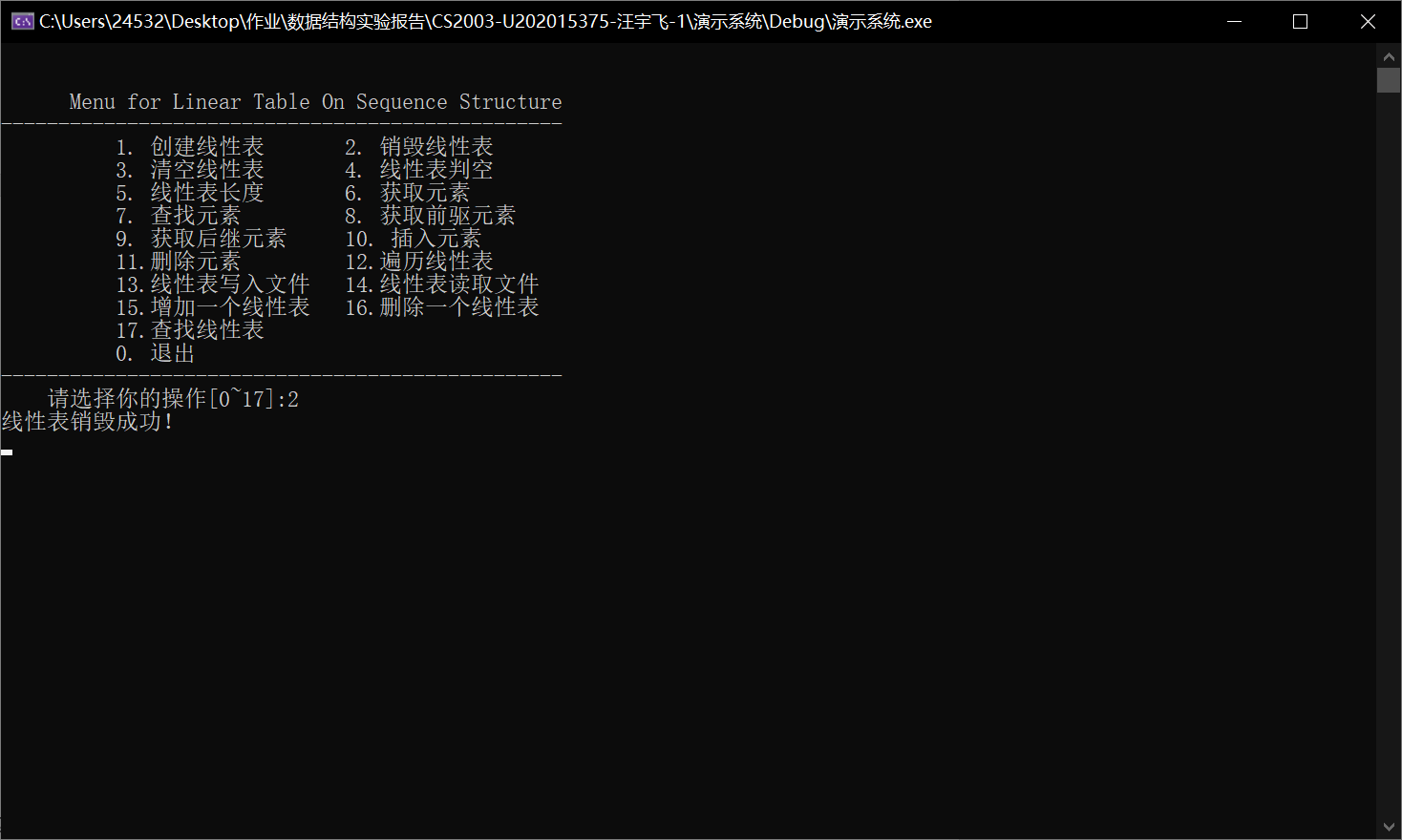


图1-7

用例：无

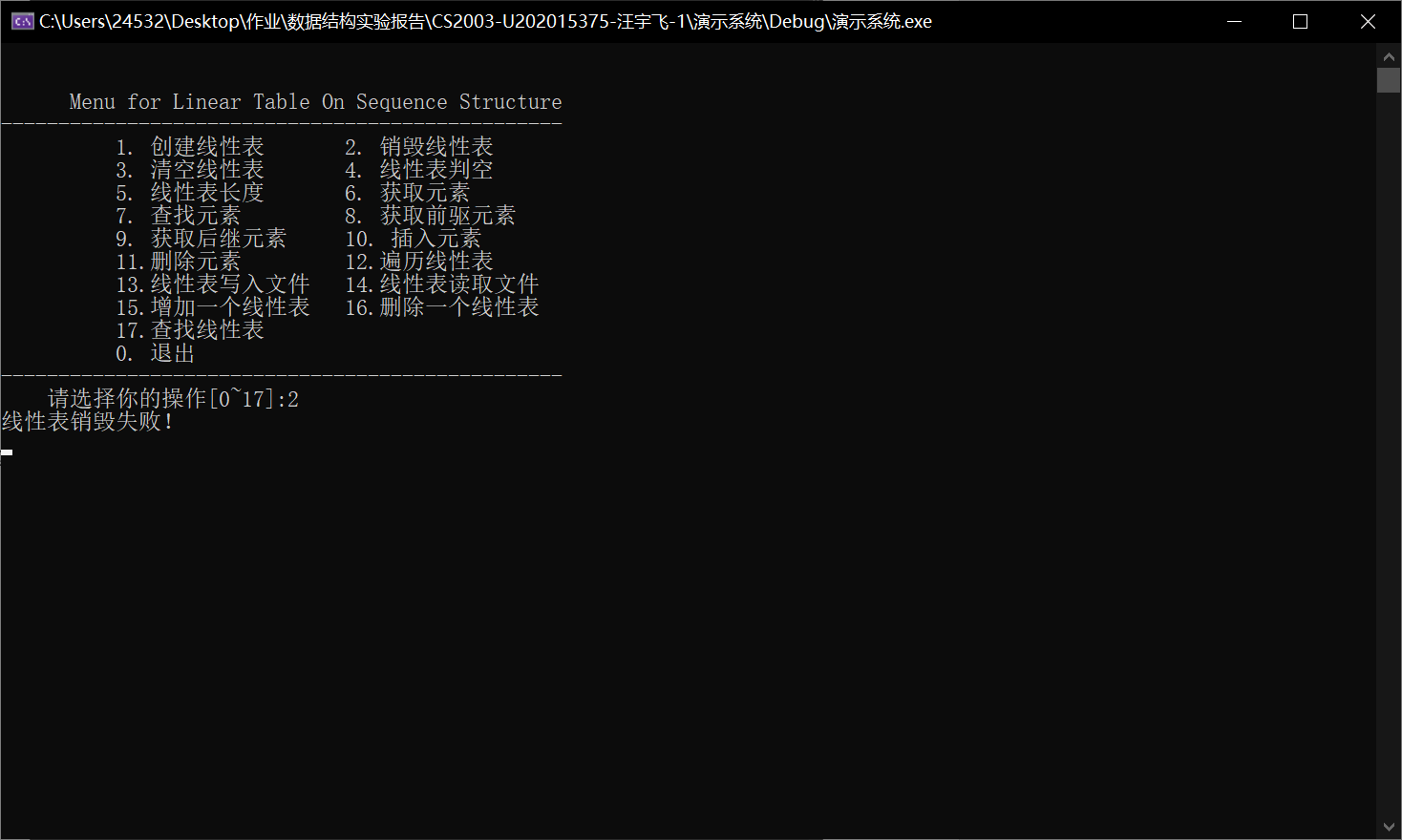
测试结果：线性表销毁失败（如图1-8）

图1-8

1. 清空线性表

用例：无

测试结果：线性表清空失败（如图1-9）

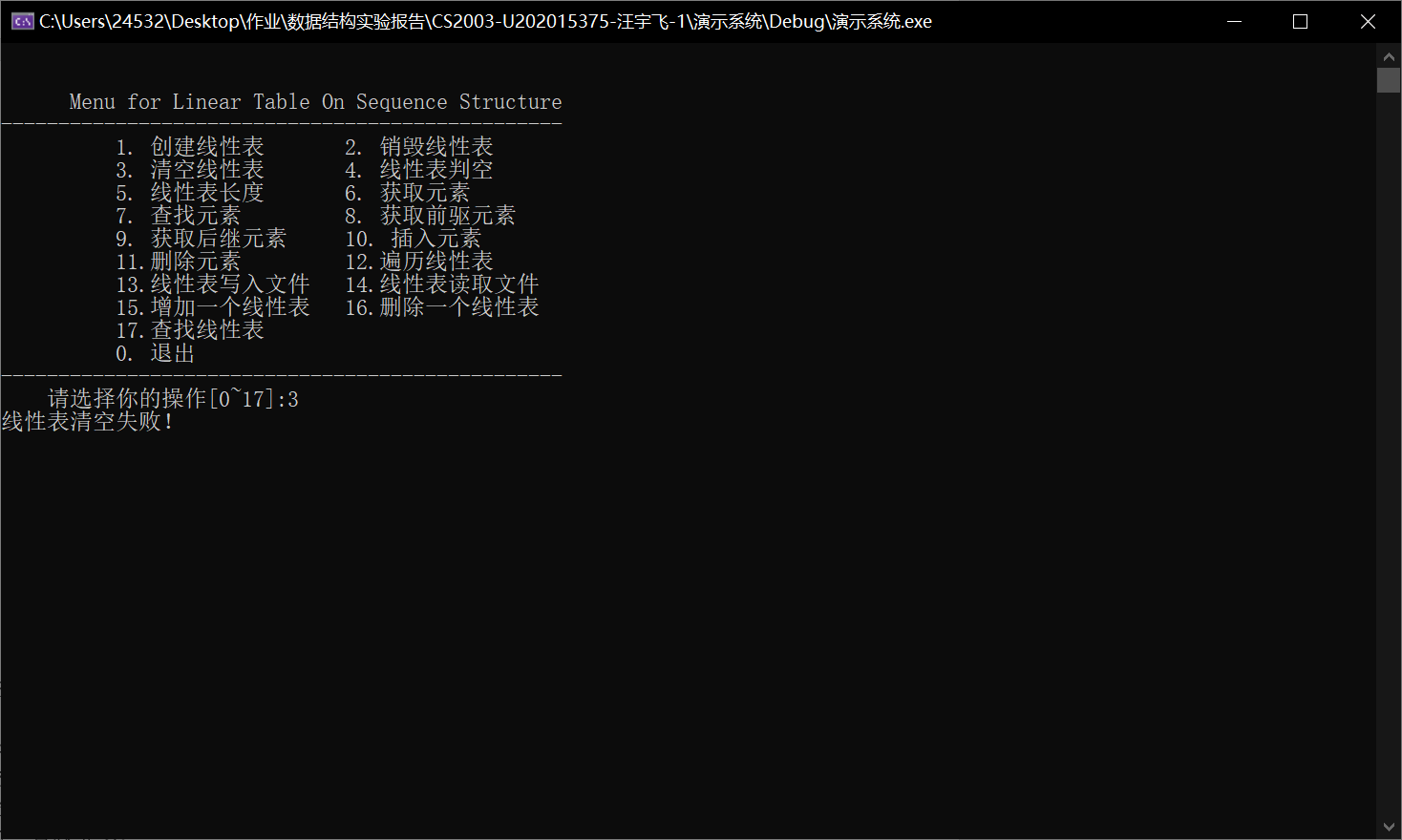


图1-9

用例：任意线性表，如（1，3，8）（2，2，2，1）等

测试结果：线性表清空成功（如图1-10）

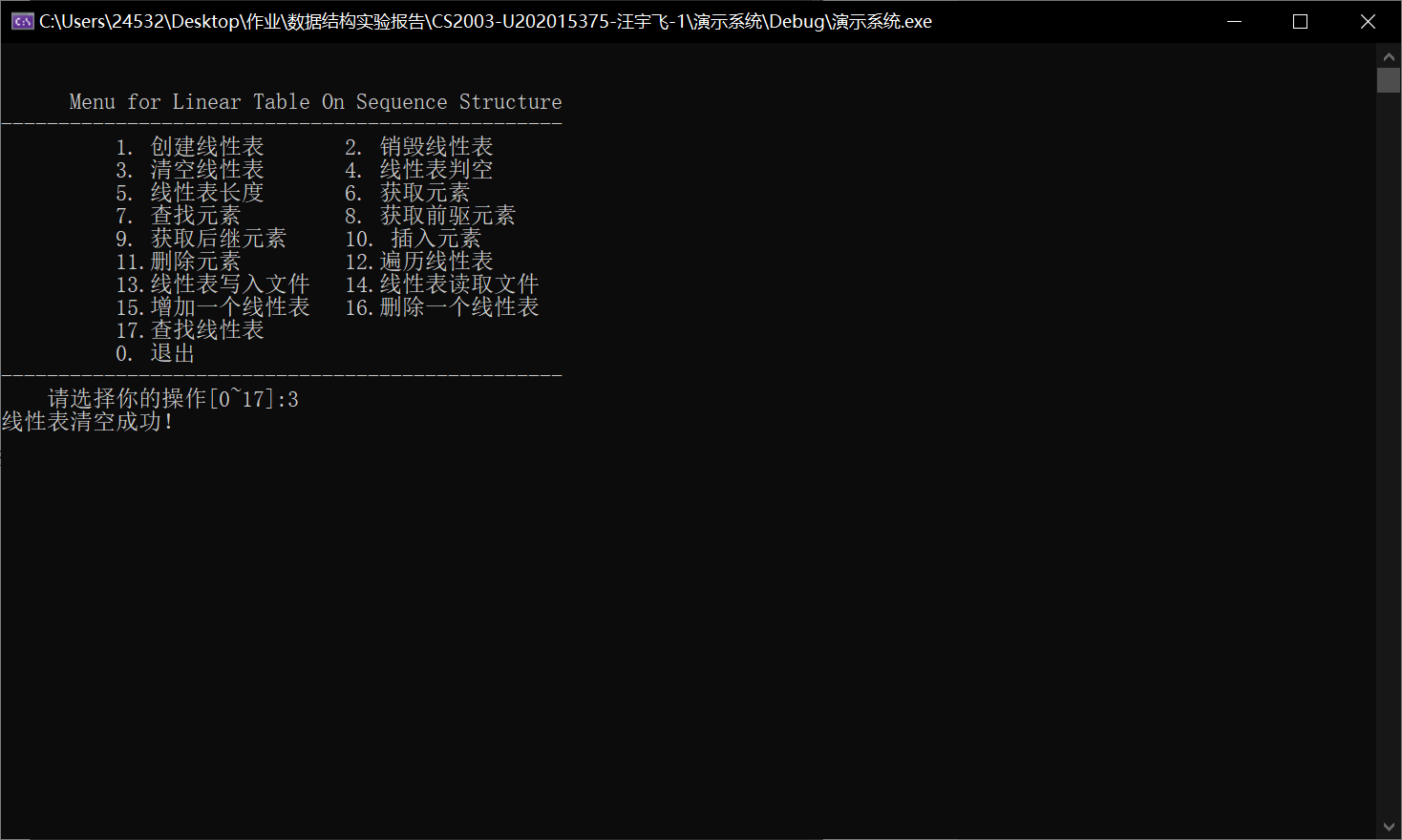


图1-10

1. 线性表判空

用例：无

测试结果：线性表不存在（如图1-11）

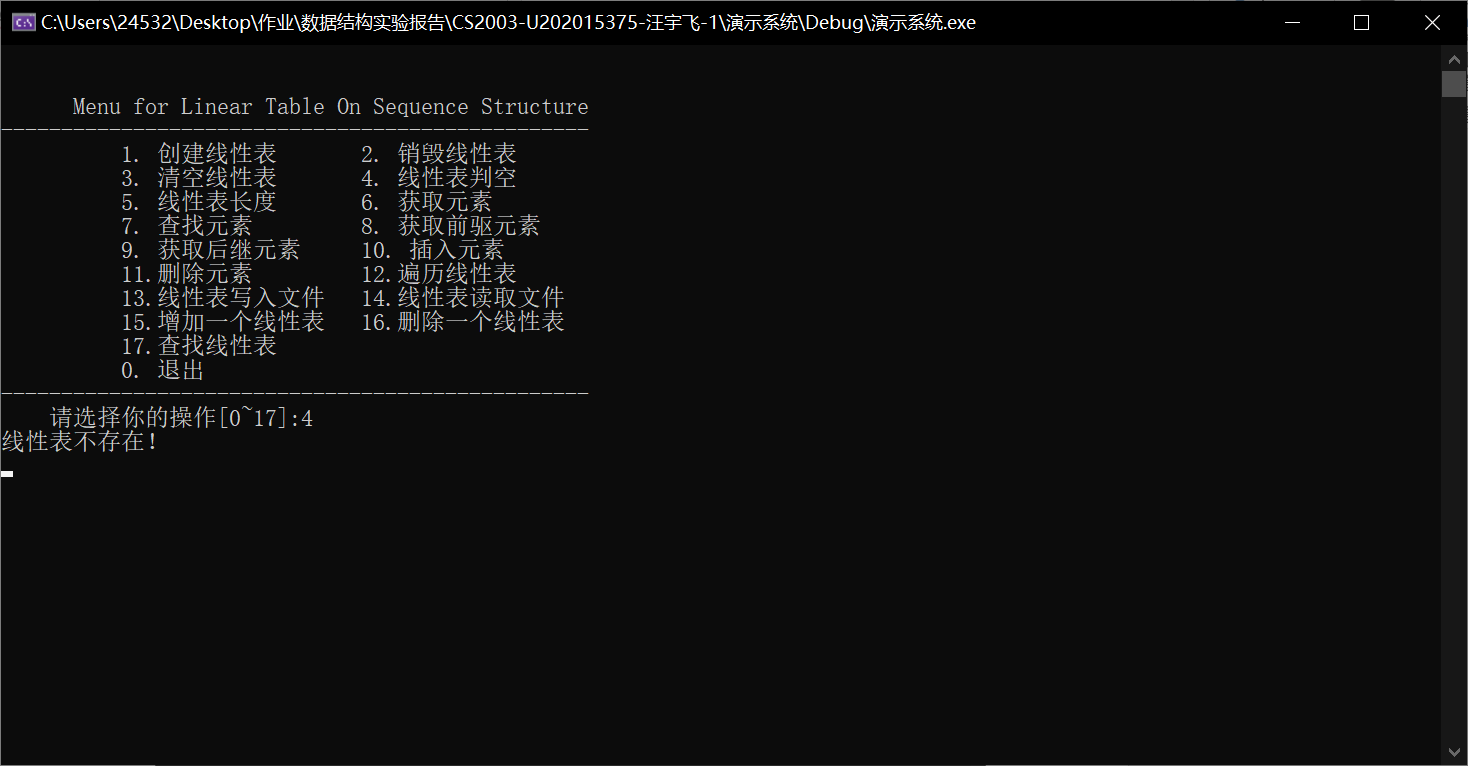


图1-11

用例：空线性表

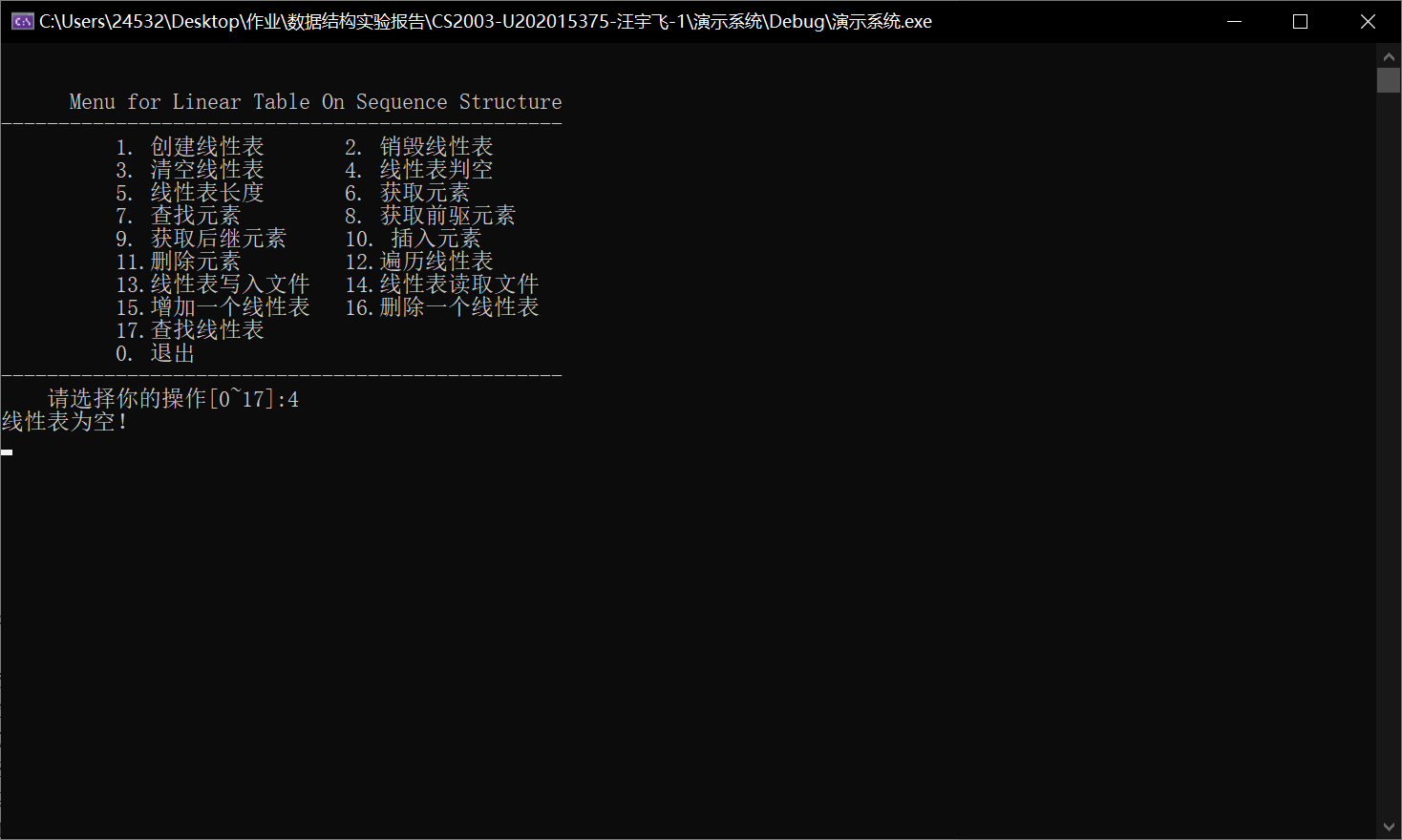
测试结果：线性表为空（如图1-12）

图1-12

用例:任意线性表，如（1，3，8）（2，2，2，1）等

测试结果：线性表不为空（如图1-13）

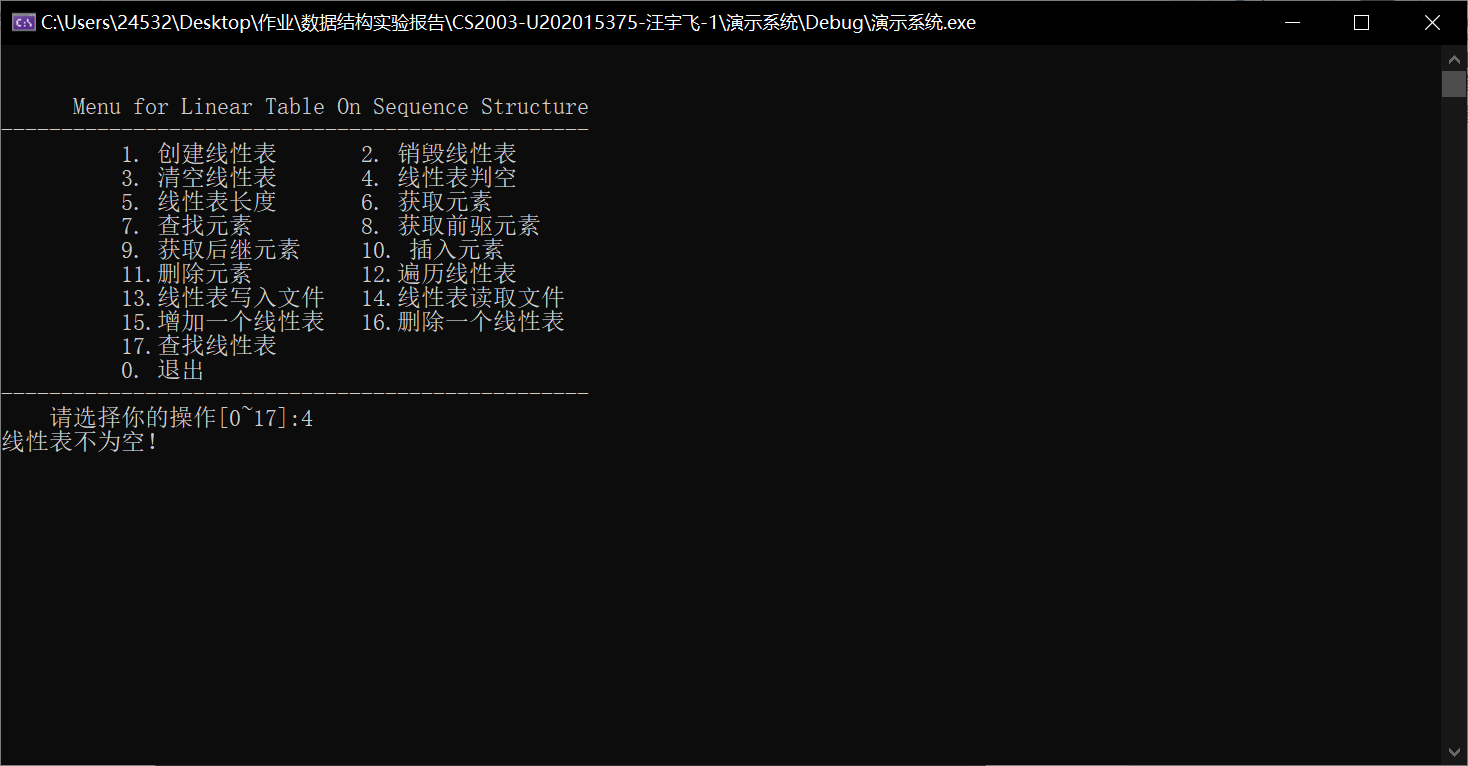


图1-13

1. 线性表长度

用例：无

测试结果：线性表不存在（如图1-14）

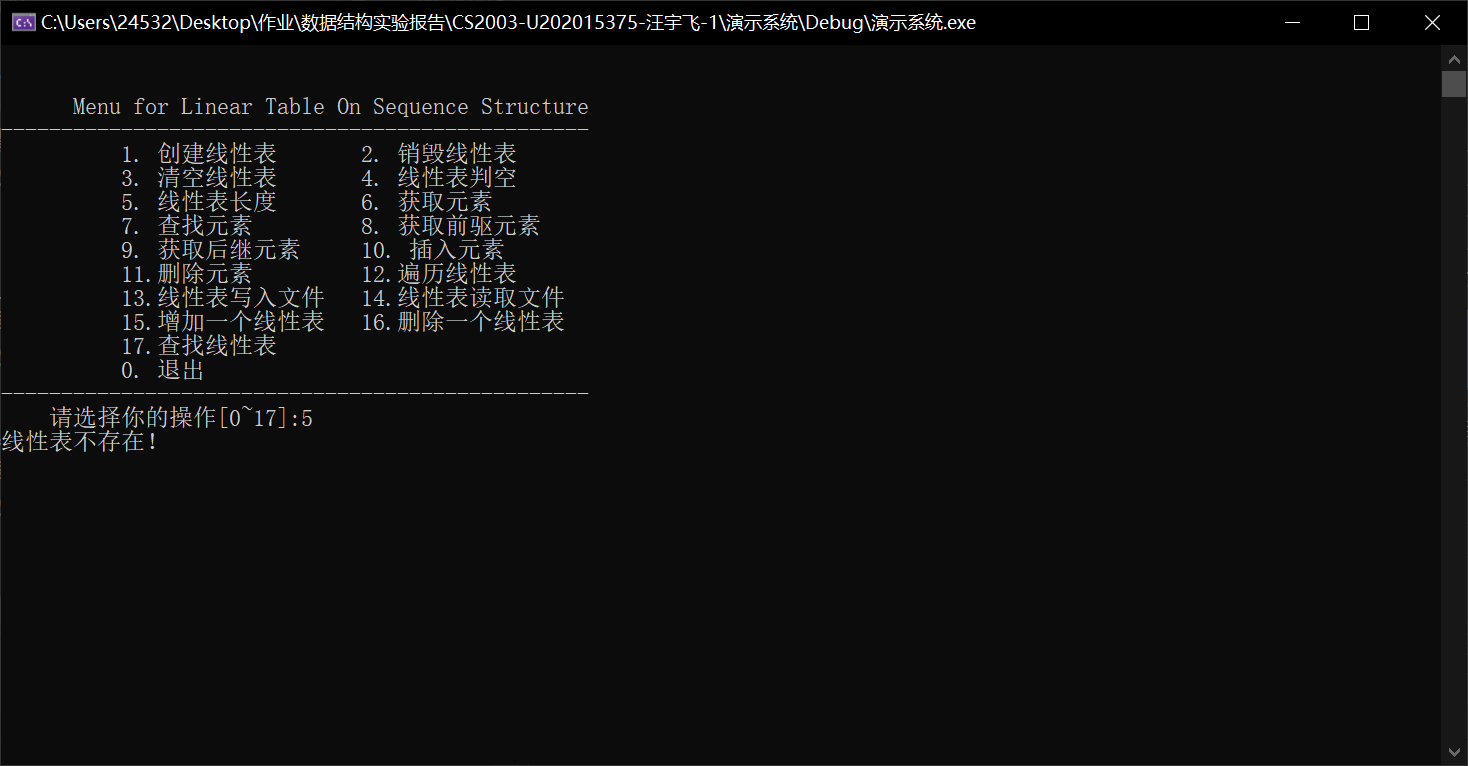


图1-14

用例：（1，2，3，4）

测试结果：4（如图1-15）



图1-15

1. 获取元素

用例：无

测试结果：线性表不存在（如图1-16）

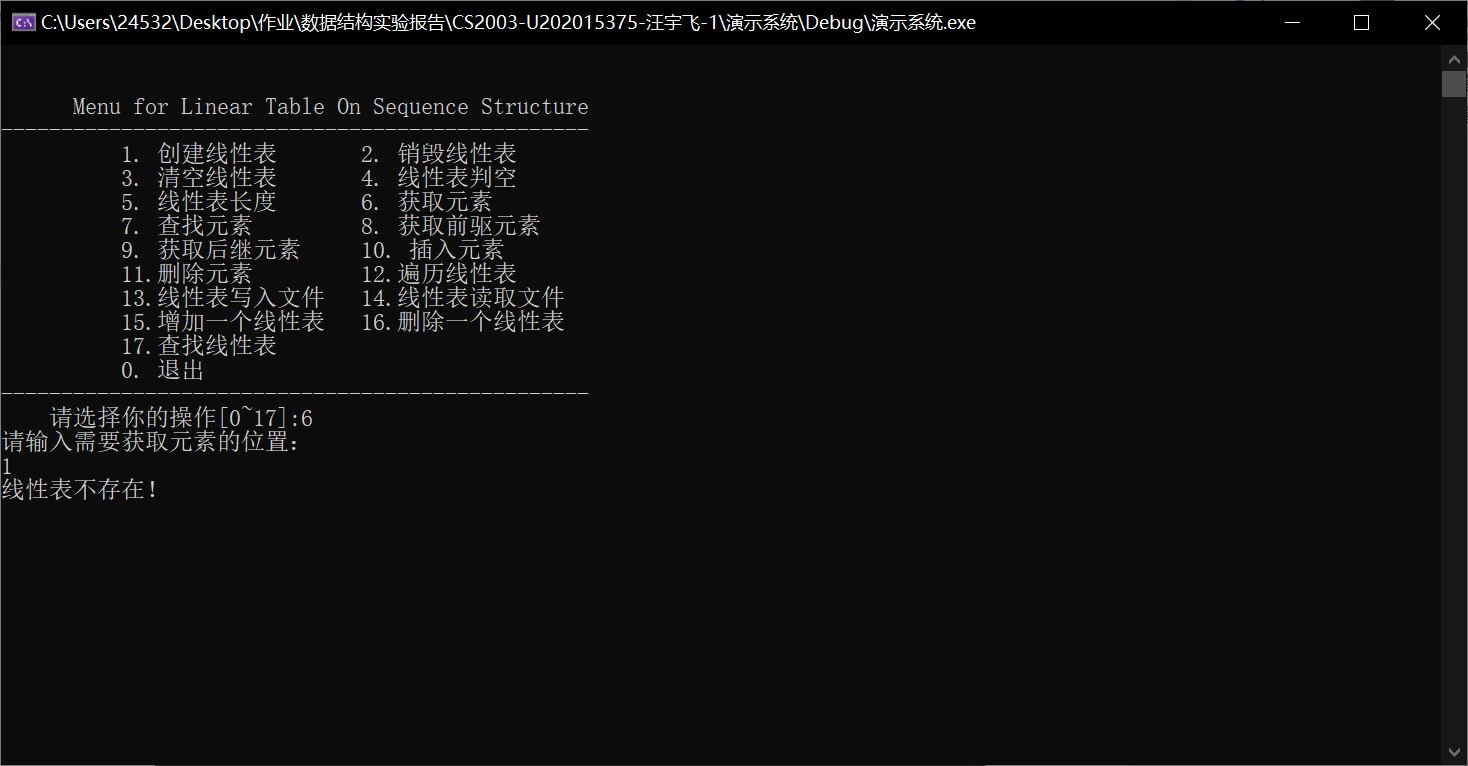


图1-16

用例：（1，2，3，4） i=2

测试结果：元素获取成功（如图1-17）

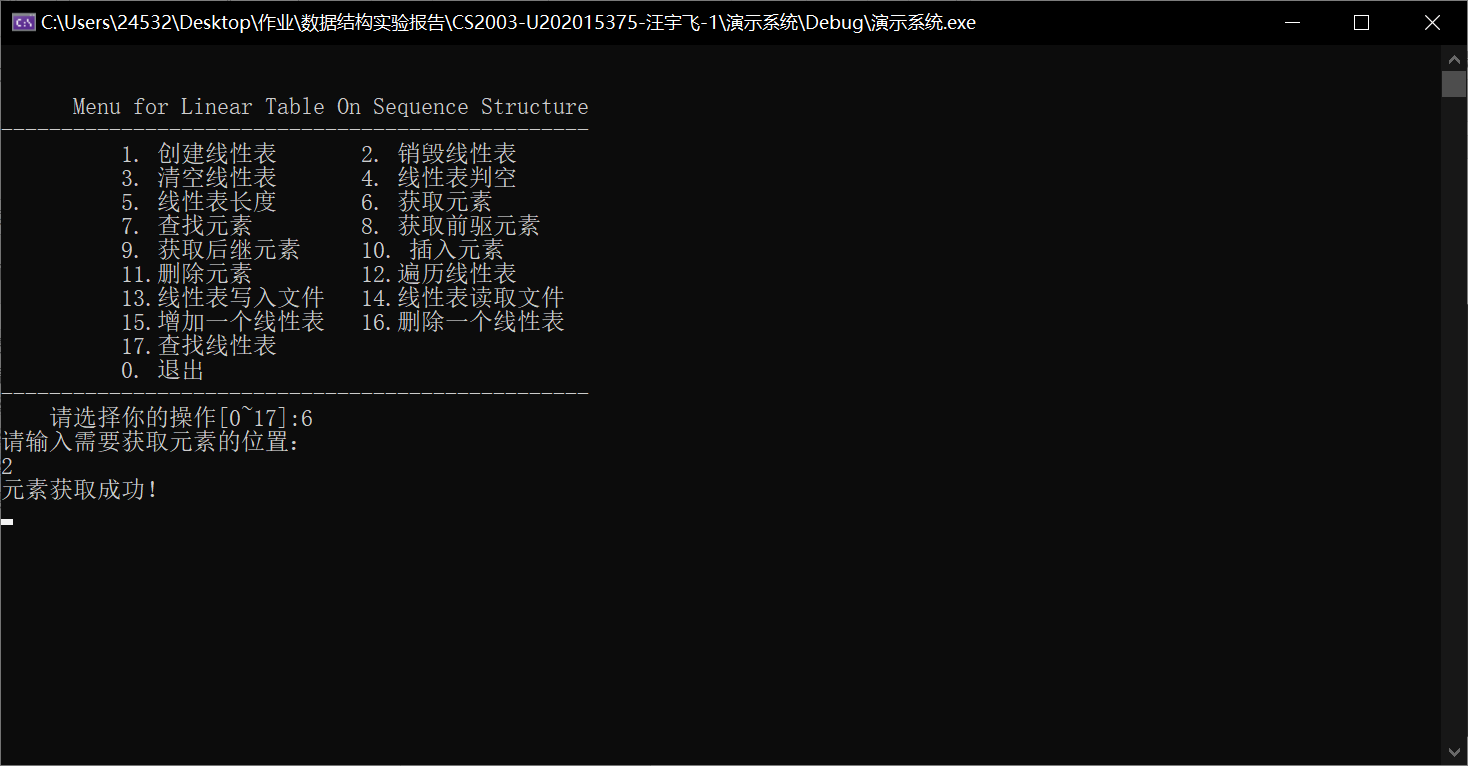


图1-17

用例：（1，2，3，4） i=6

测试结果：i不合法（如图1-18）

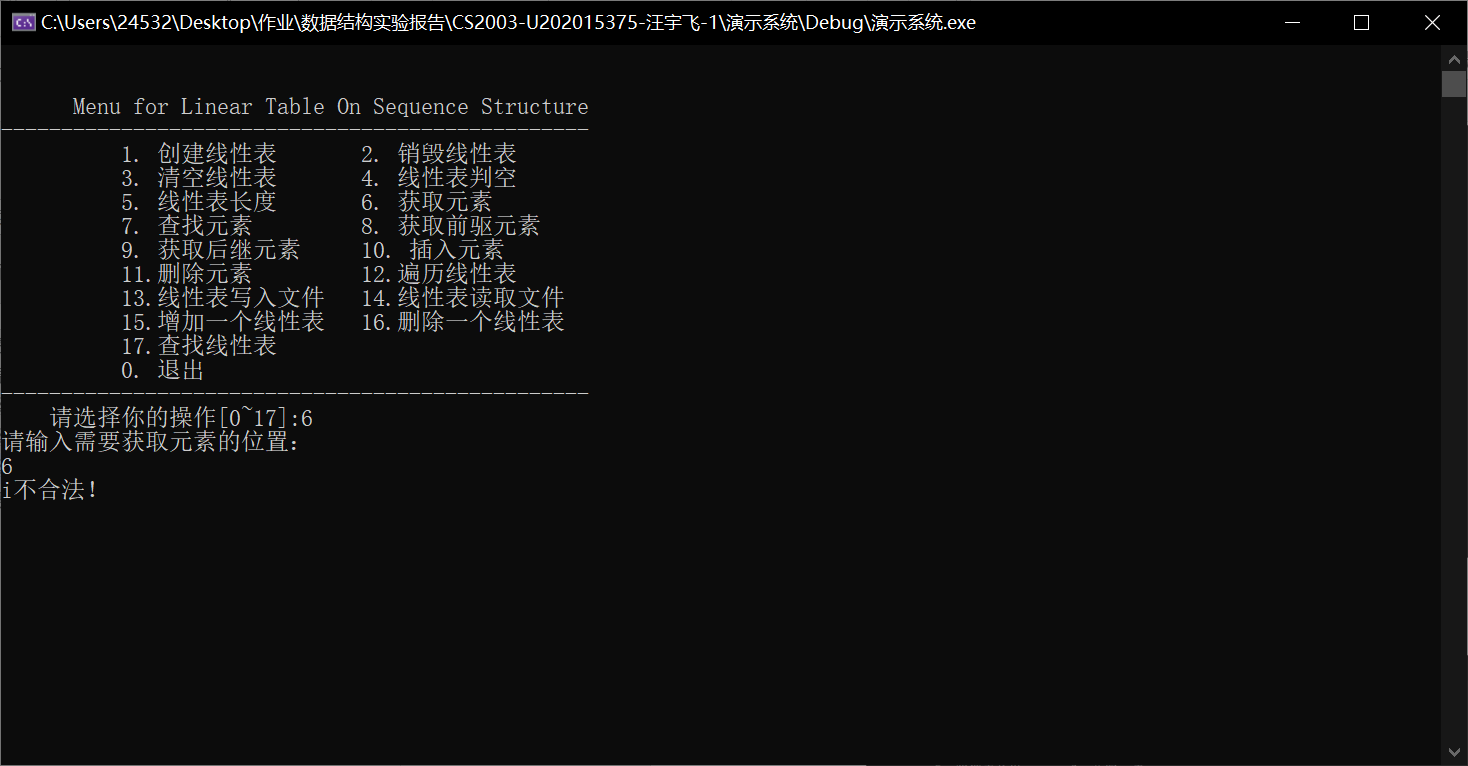


图1-18

1. 查找元素

用例：无

测试结果：线性表不存在（如图1-19）



图1-19

用例（1，2，3，4） e=2

测试结果：该元素是第2个元素（如图1-20）



图1-20

用例（1，2，3，4） e=8

测试结果：未找到该元素（如图1-21）

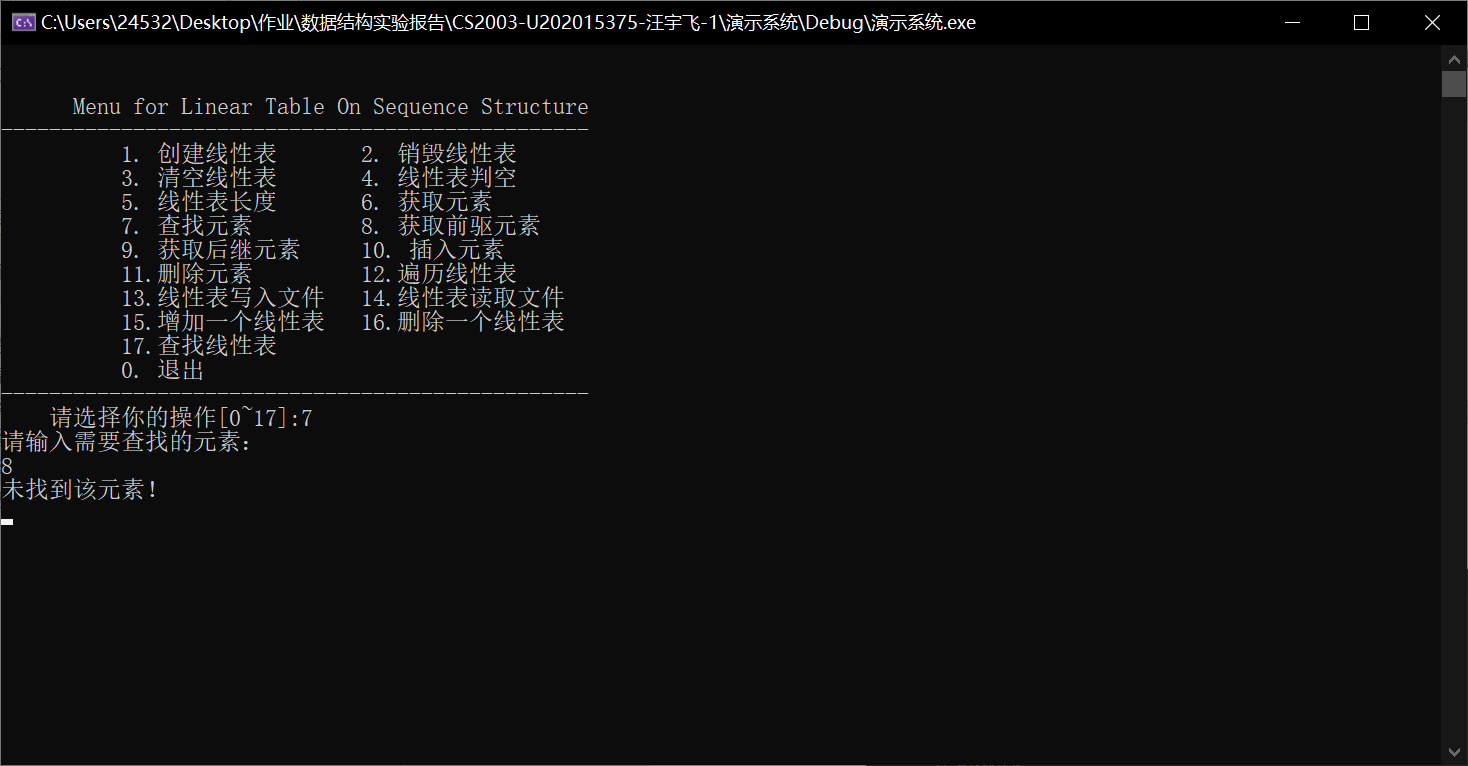


图1-21

1. 获取前驱元素

用例：无

测试结果：线性表不存在（如图1-22）

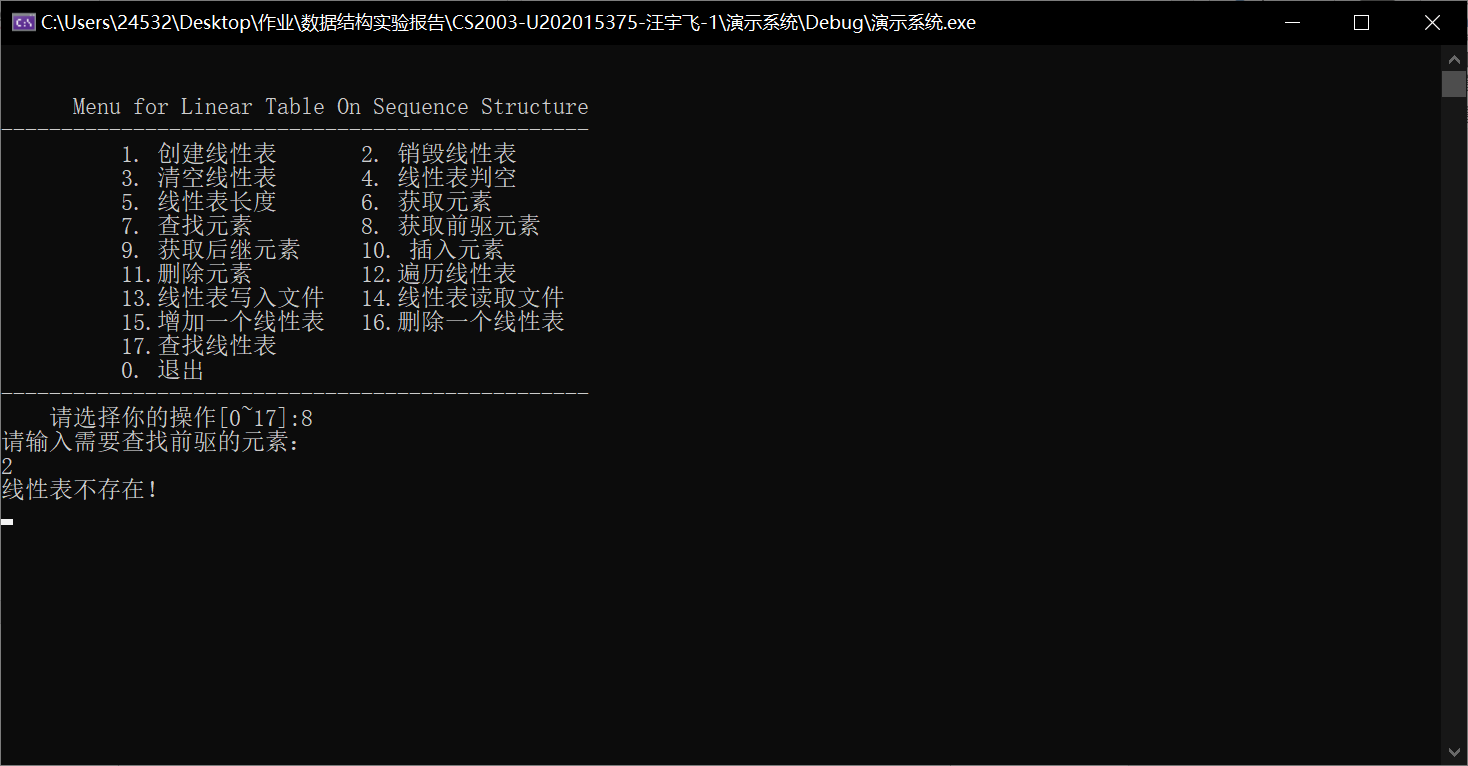


图1-22

用例：（1，2，3，4） e=3

测试结果：前驱为2（如图1-23）



图1-23

用例：（1，2，3，4） e=5

测试结果：前驱不存在（如图1-24）



图1-24

1. 获取后继元素

用例：无

测试结果：线性表不存在（如图1-25）

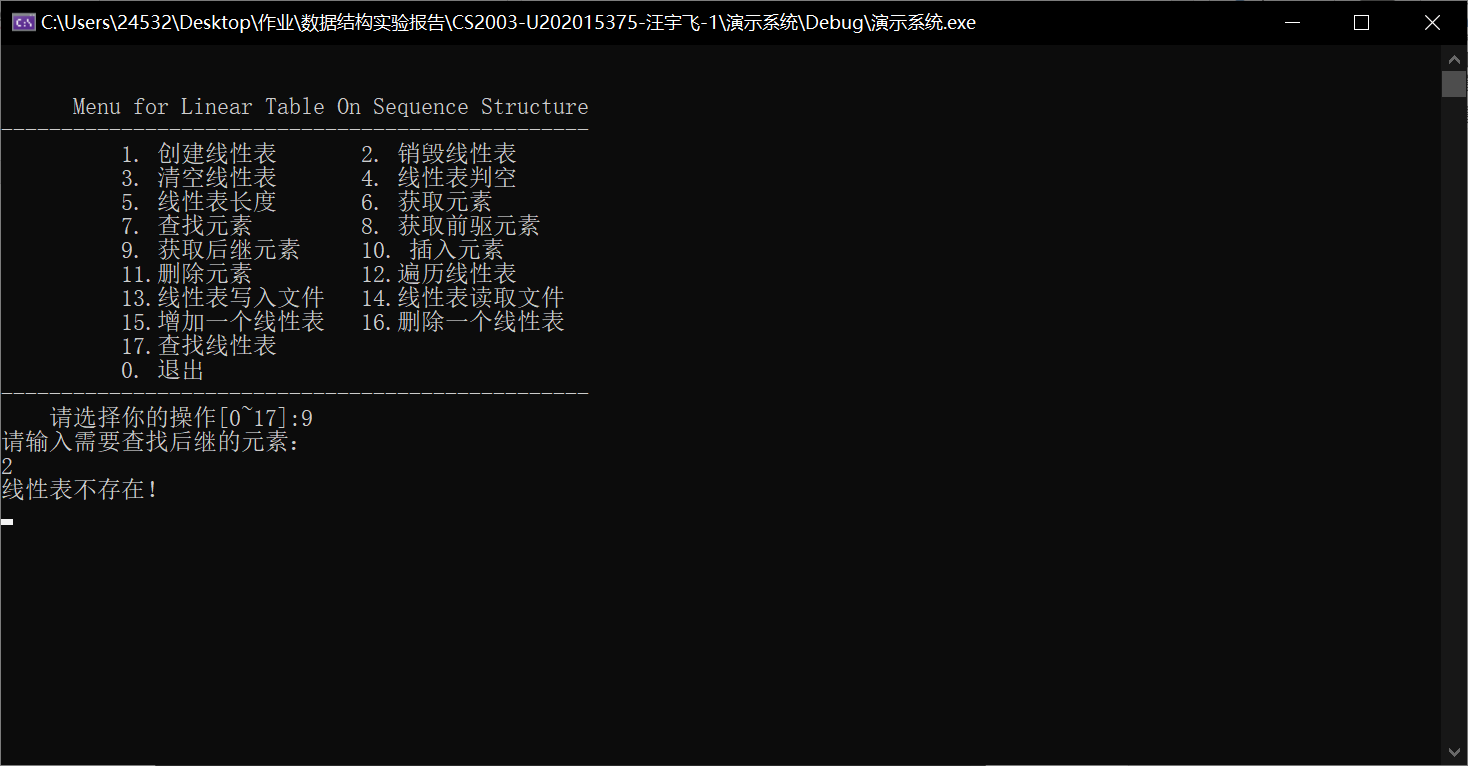


图1-25

用例：（1，2，3，4） e=2

测试结果：后继为3（如图1-26）

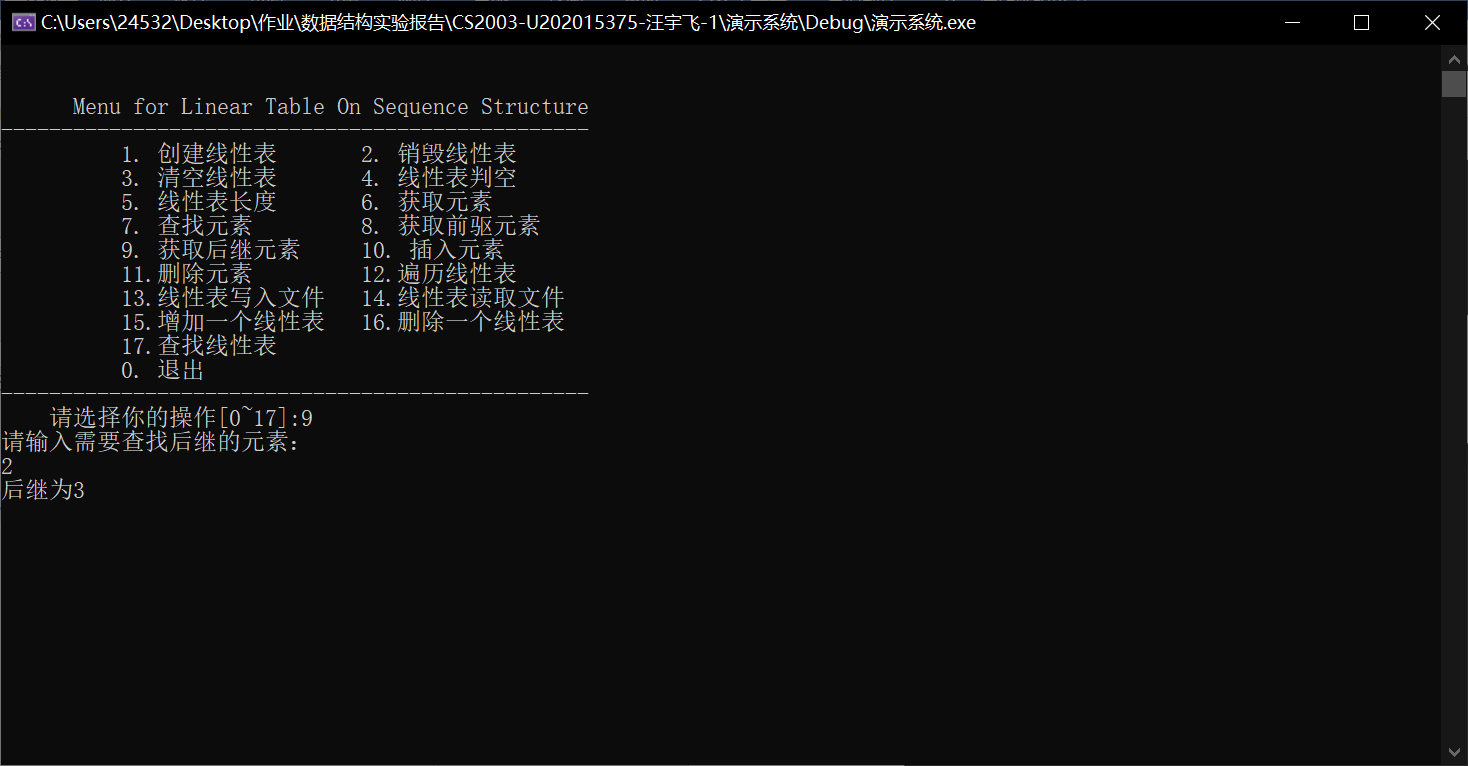


图1-26

用例：（1，2，3，4） e=4

测试结果：后继不存在（如图1-27）

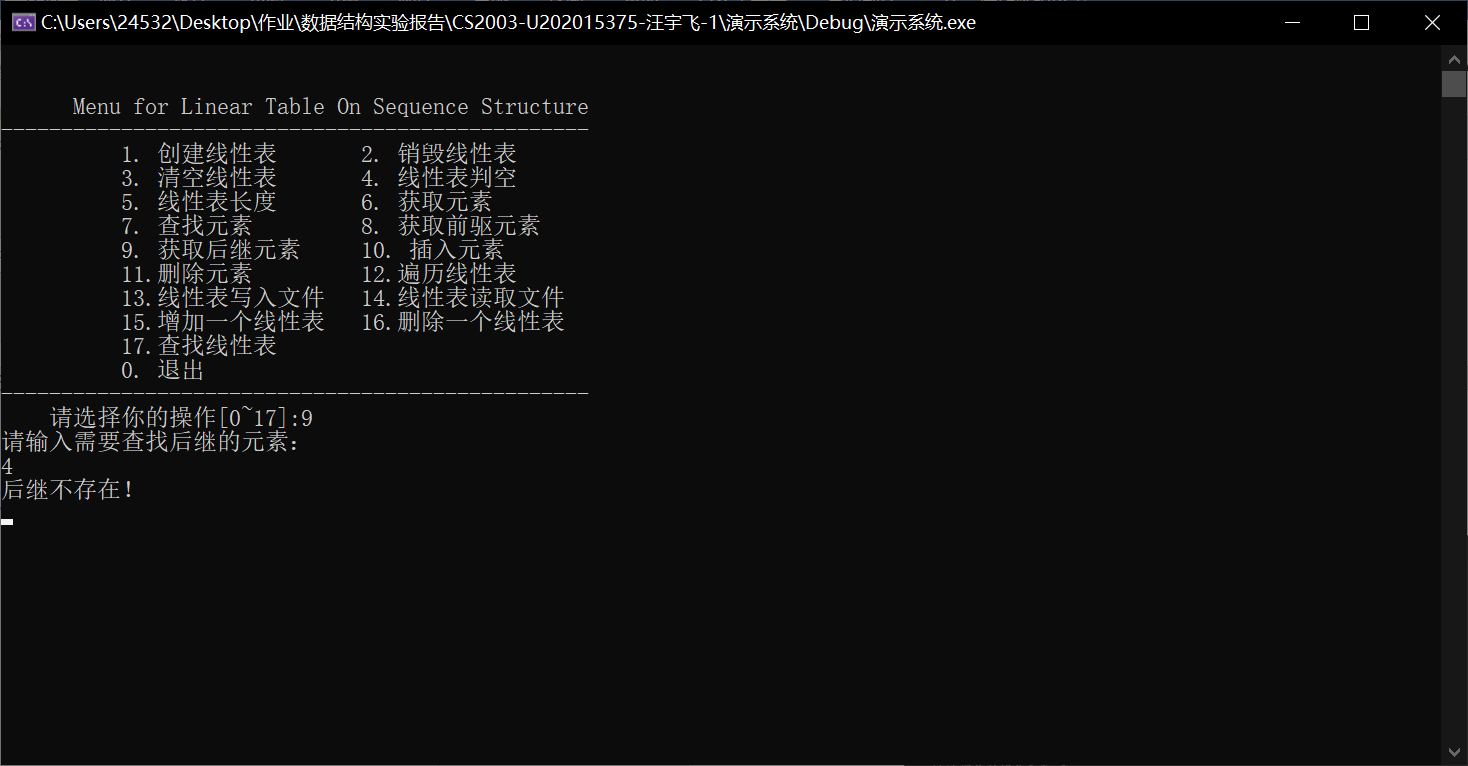


图1-27

1. 插入元素

用例：无

测试结果：线性表不存在（如图1-28）



图1-28

用例：（1，2，3，4） e=1 i=0

测试结果：插入位置错误（如图1-29）

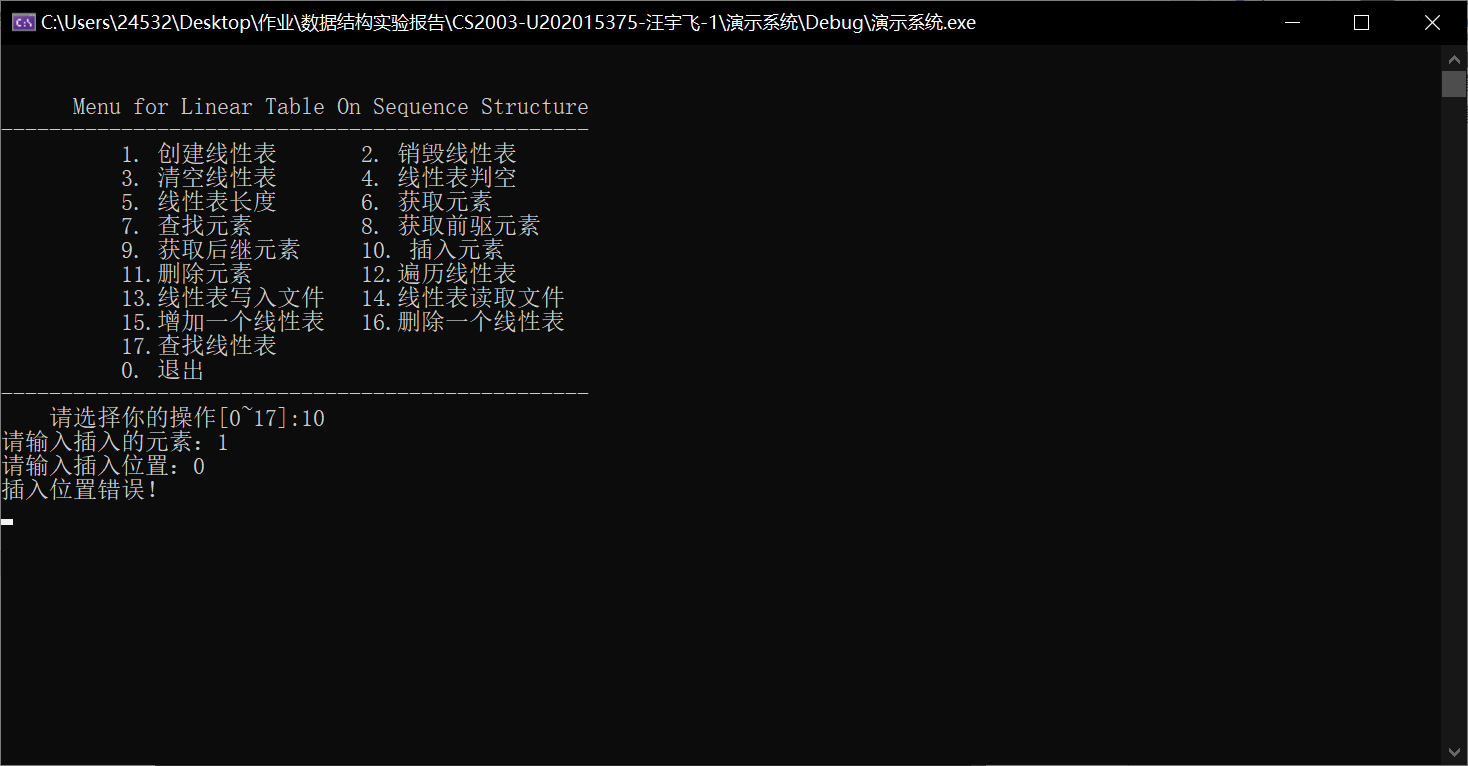


图1-29

用例：（1，2，3，4） e=1 i=1

测试结果：已插入成功（如图1-30）

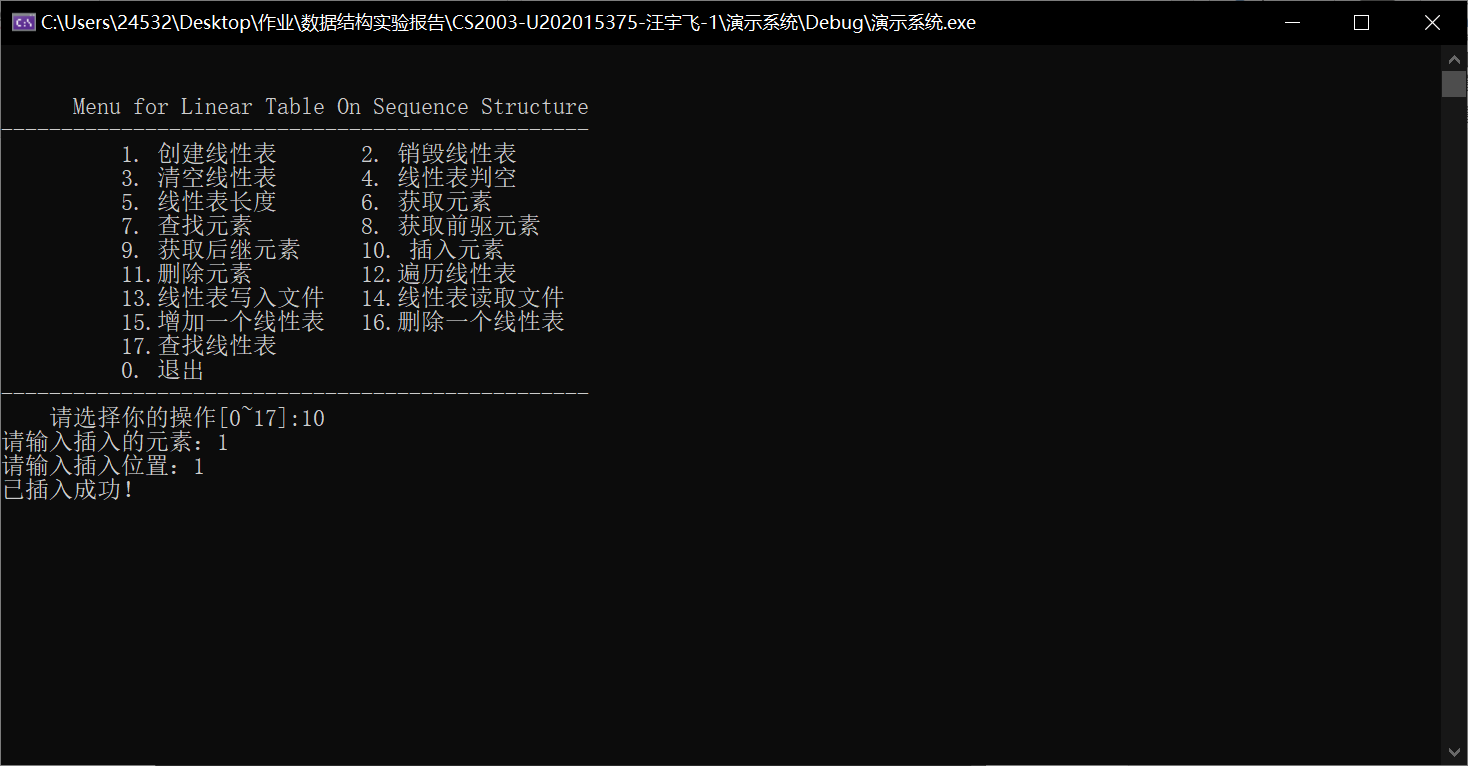


图1-30

1. 删除元素

用例：无

测试结果：线性表不存在（如图1-31）

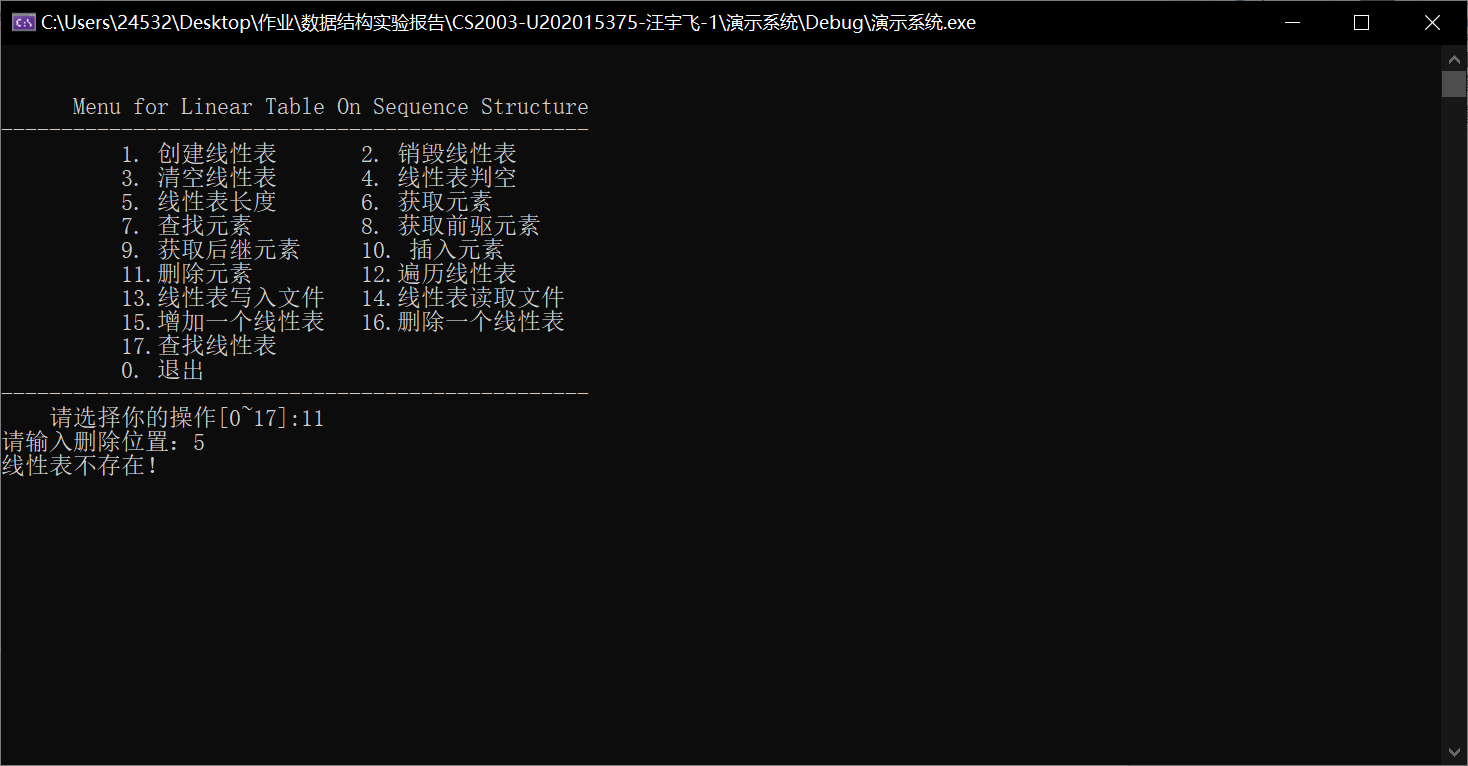


图1-31

用例：（1，2，3，4） i=5

测试结果：删除位置错误（如图1-32）

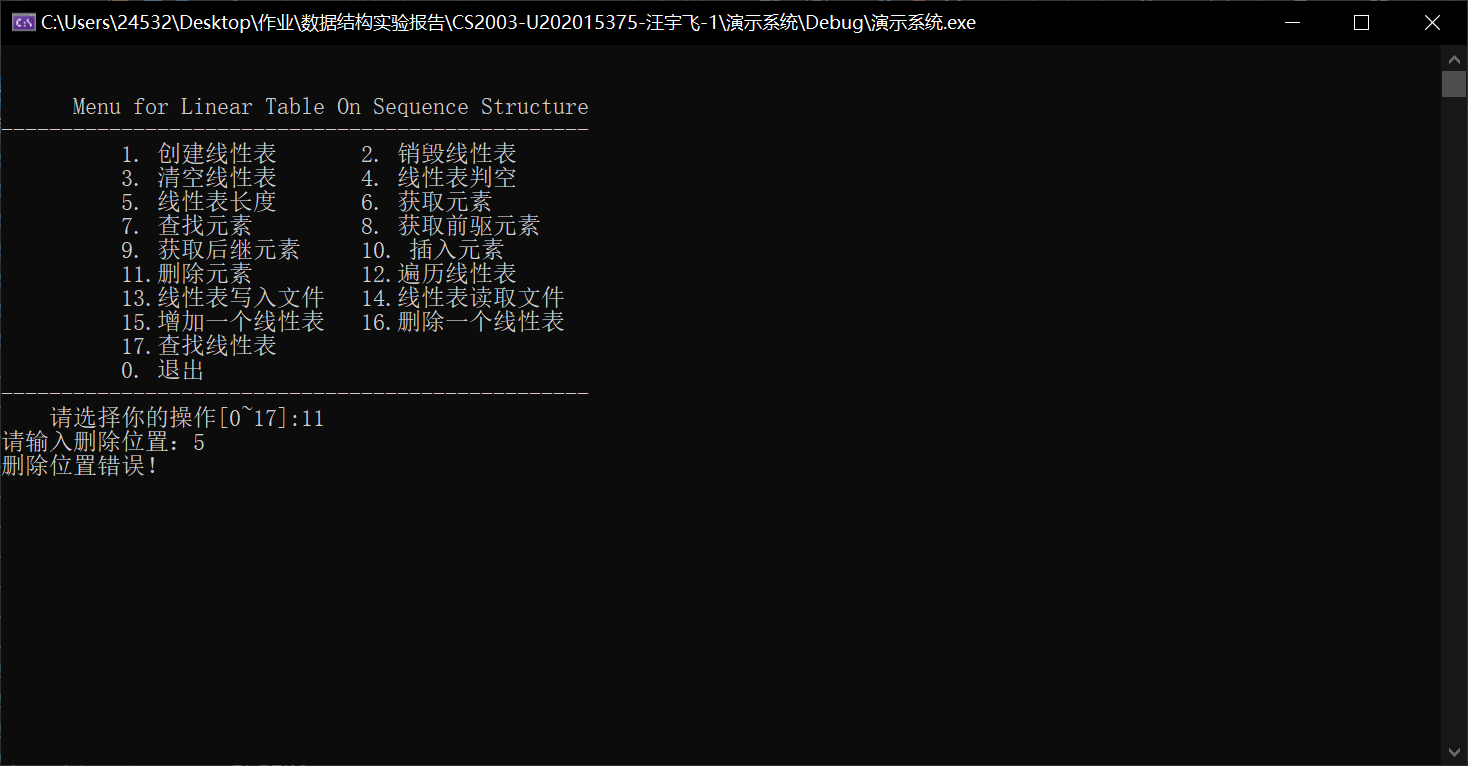


图1-32

用例：（1，2，3，4） i=3

测试结果：已删除成功（如图1-33）

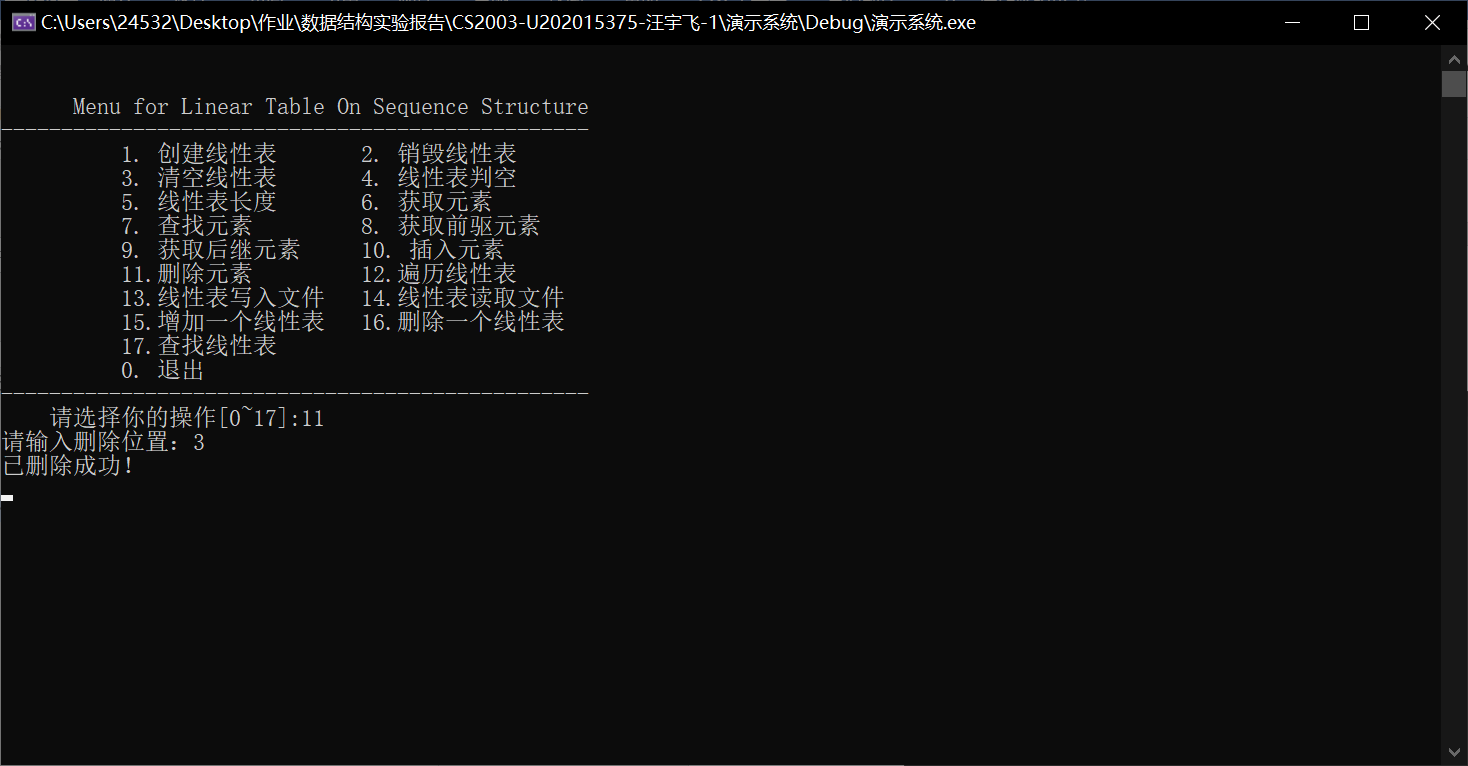


图1-33

1. 遍历线性表

用例：无

测试结果：线性表不存在（如图1-34）



图1-34

用例：（1，2，3，4）

测试结果：1 2 3 4（如图1-35）

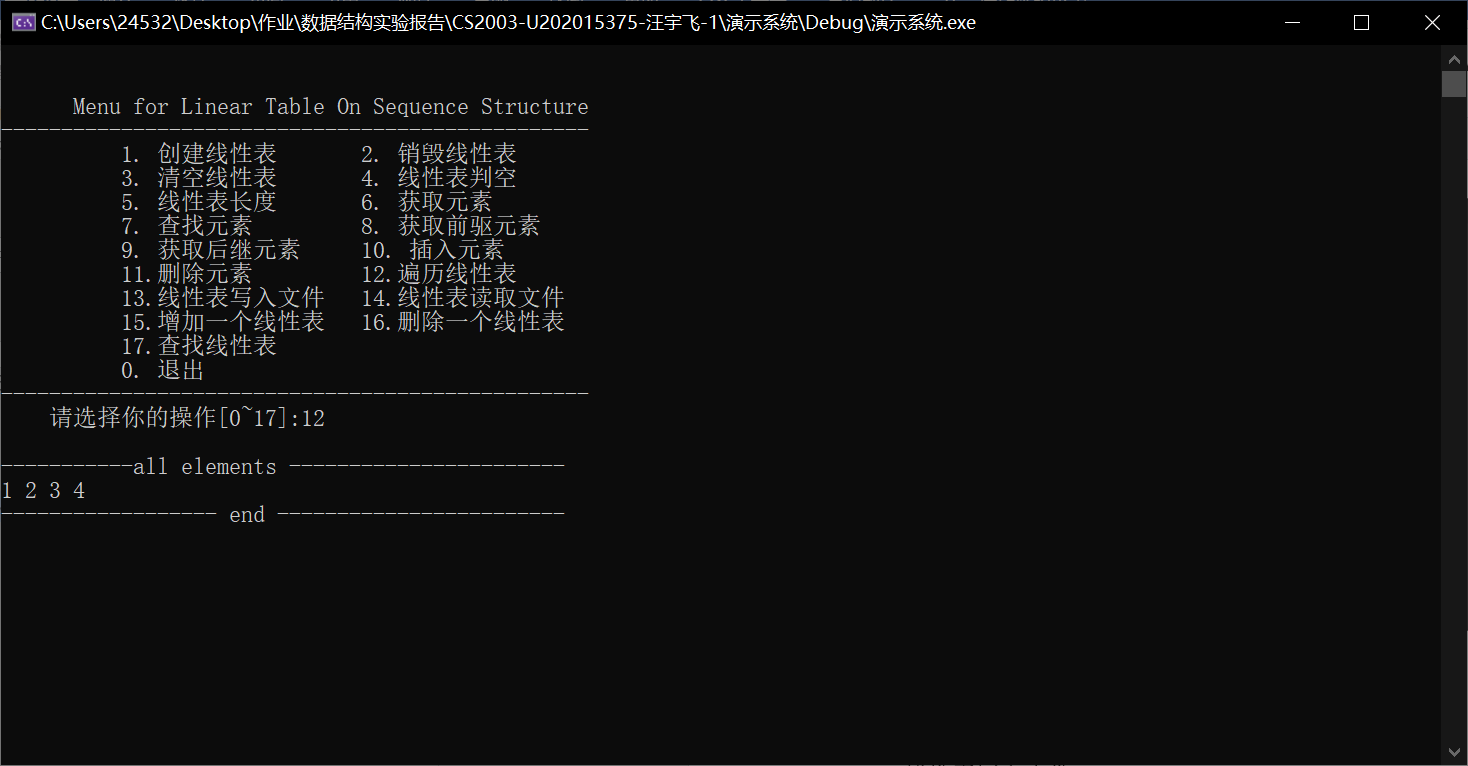


图1-35

1. 线性表写入文件

用例：无

测试结果：线性表不存在（如图1-36）

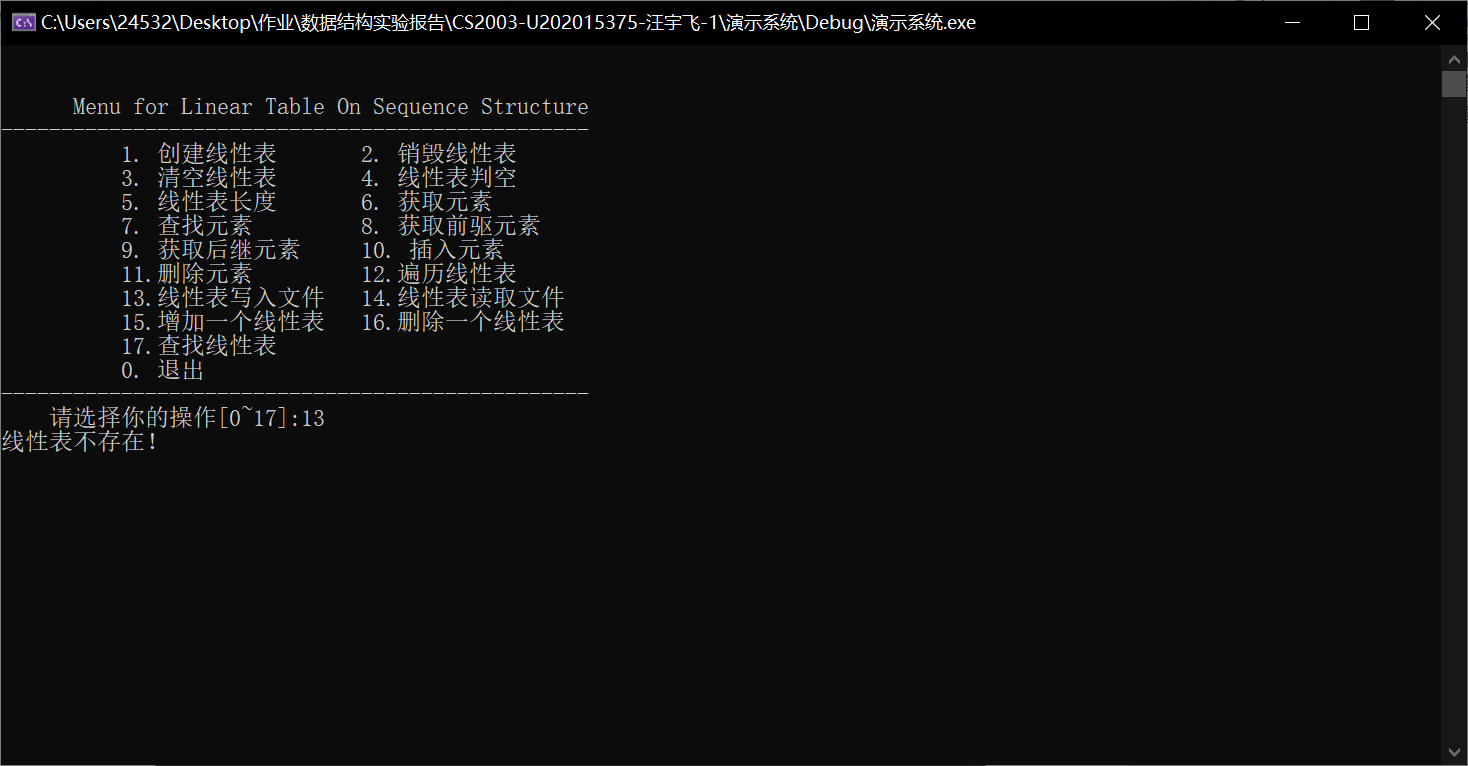


图1-36

用例：（1，2，3，4）

测试结果：线性表元素已写入文件（如图1-37）

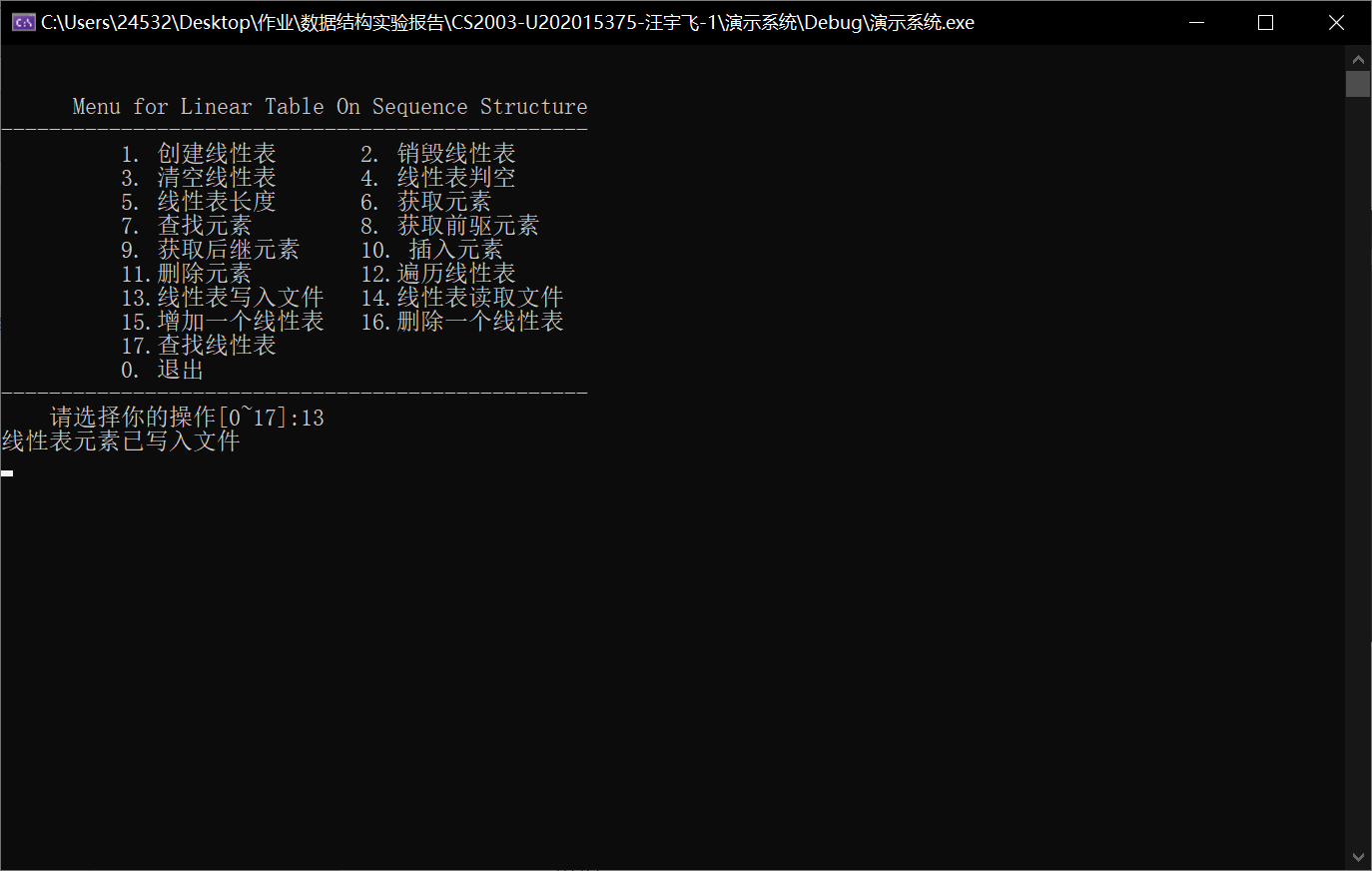


图1-37

1. 线性表读取文件

用例：任意线性表

测试结果：线性表已存在（如图1-38）

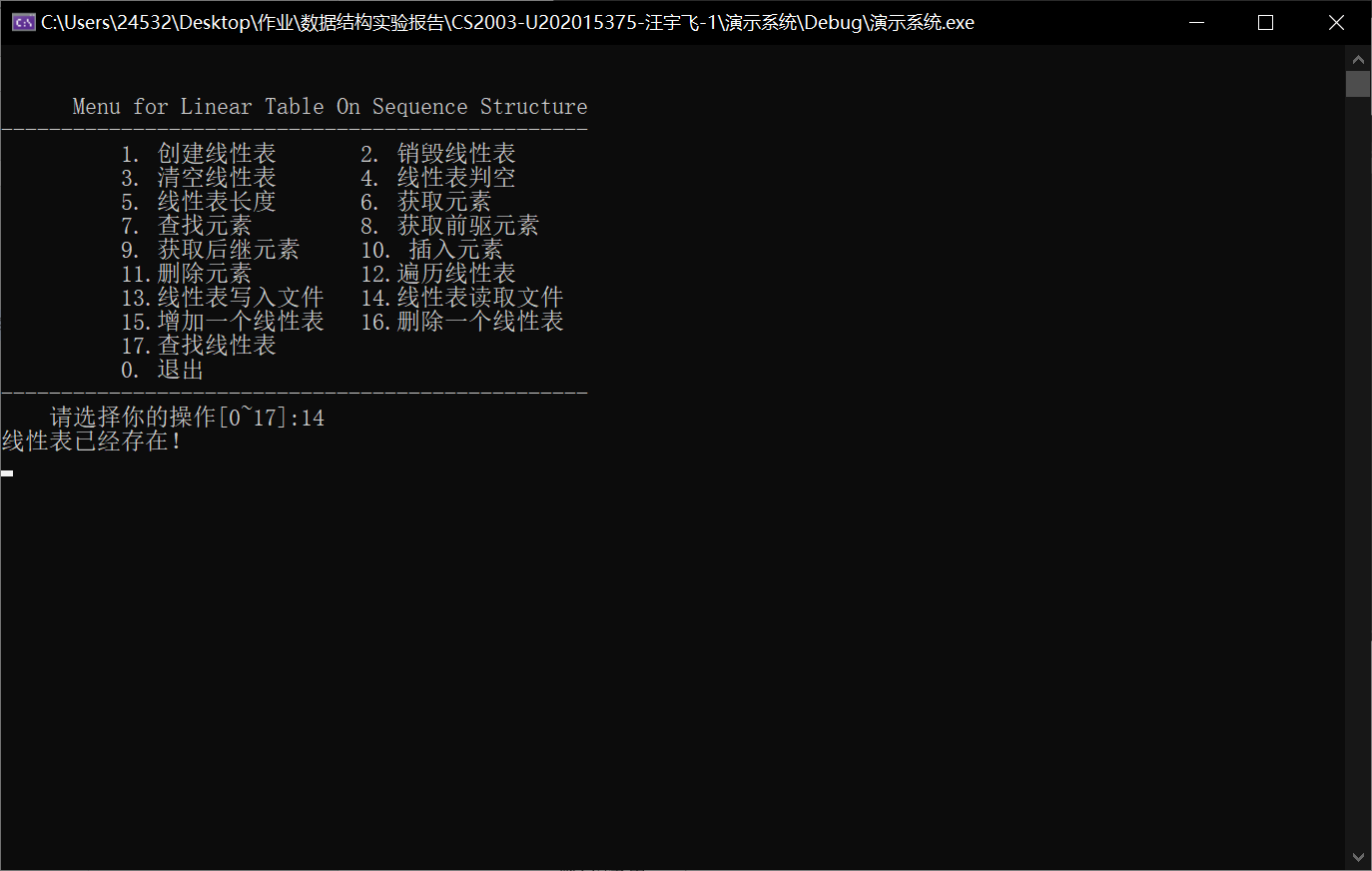


图1-38

用例：无

测试结果：线性表已读取文件（如图1-39）

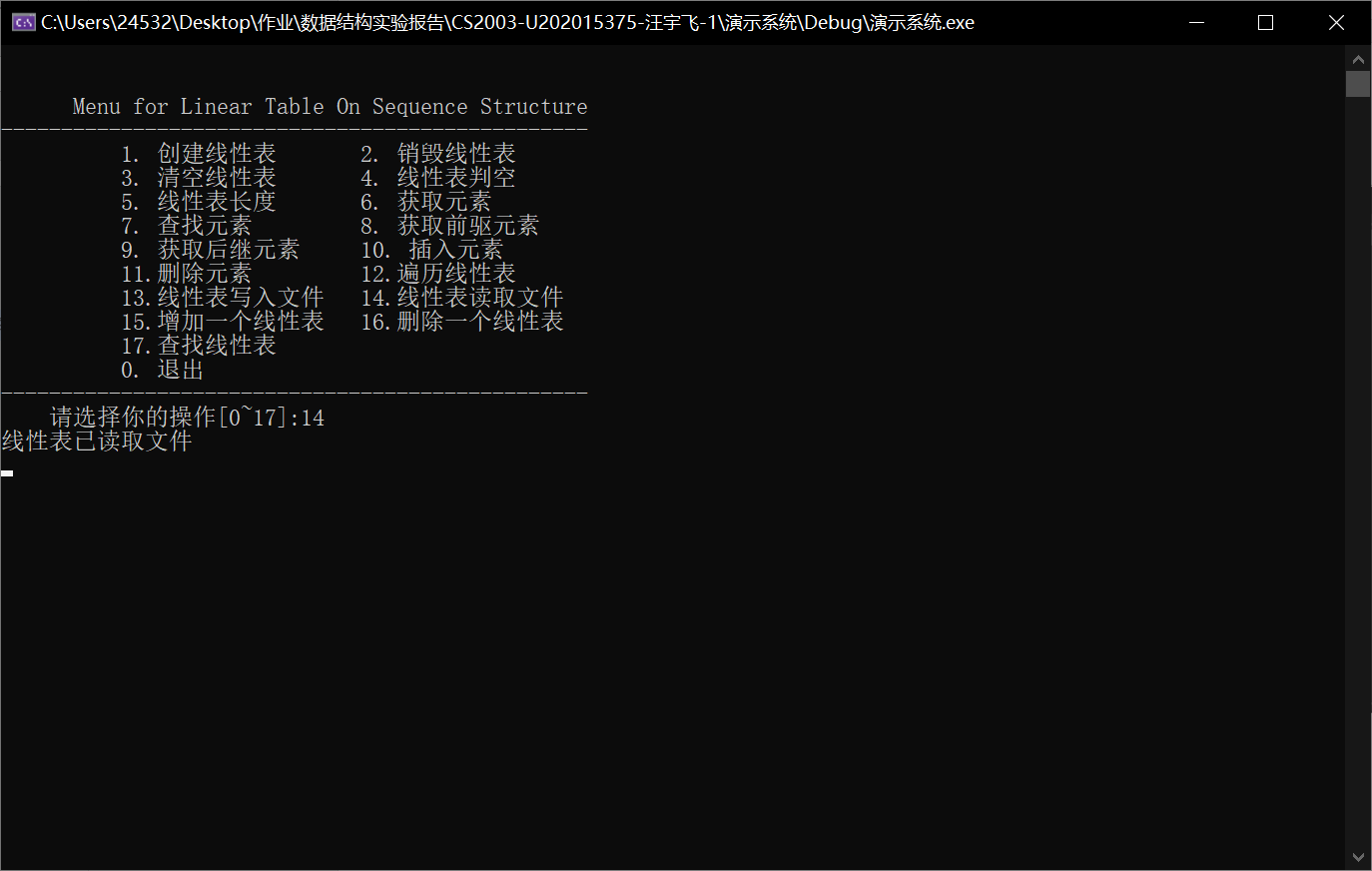


图1-39

1. 增加一个线性表

用例：“新线性表”

测试结果：线性表创建成功（如图1-40）

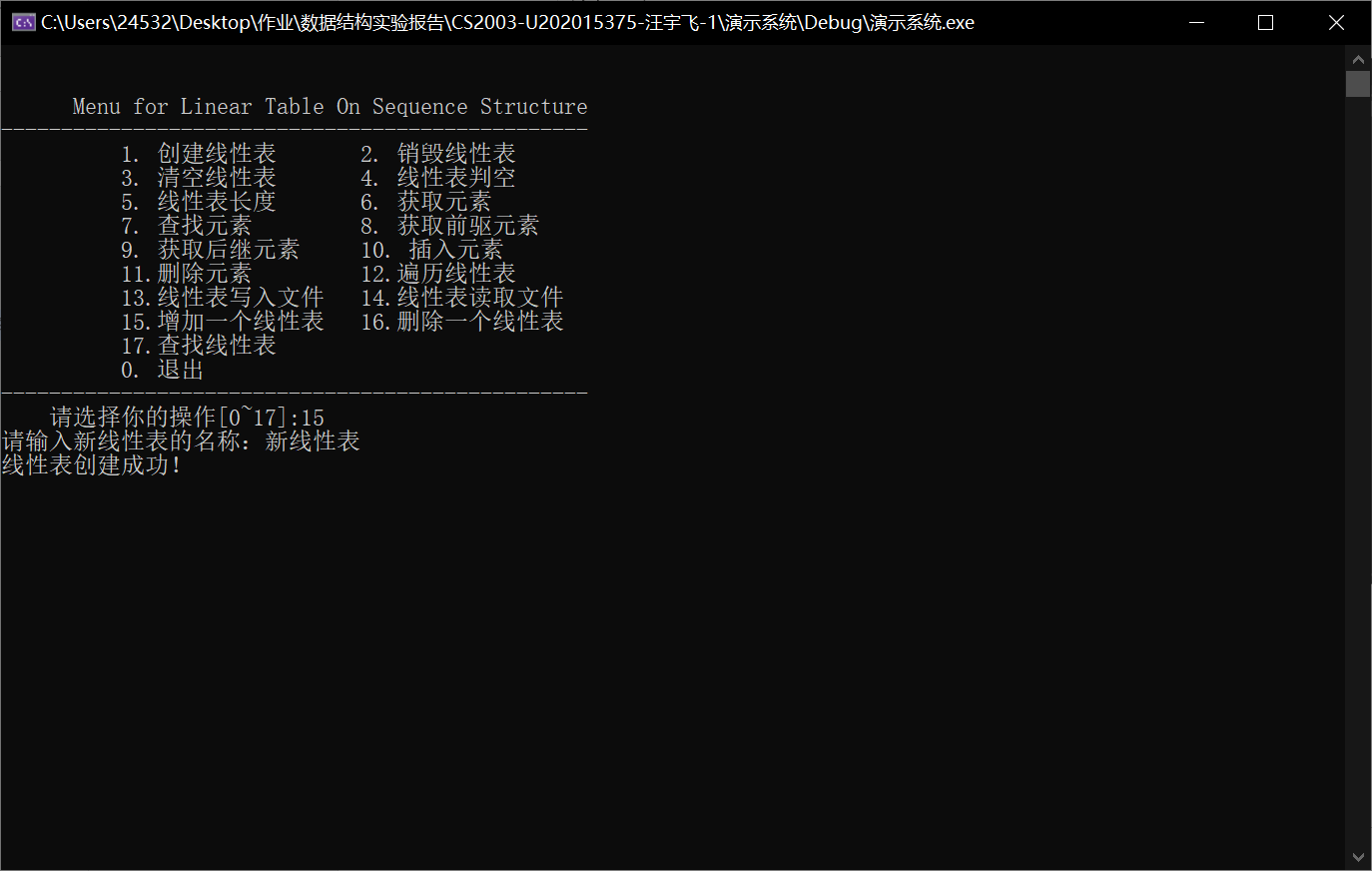


图1-40

1. 删除一个线性表

用例：“新线性表”

测试结果：删除成功（如图1-41）

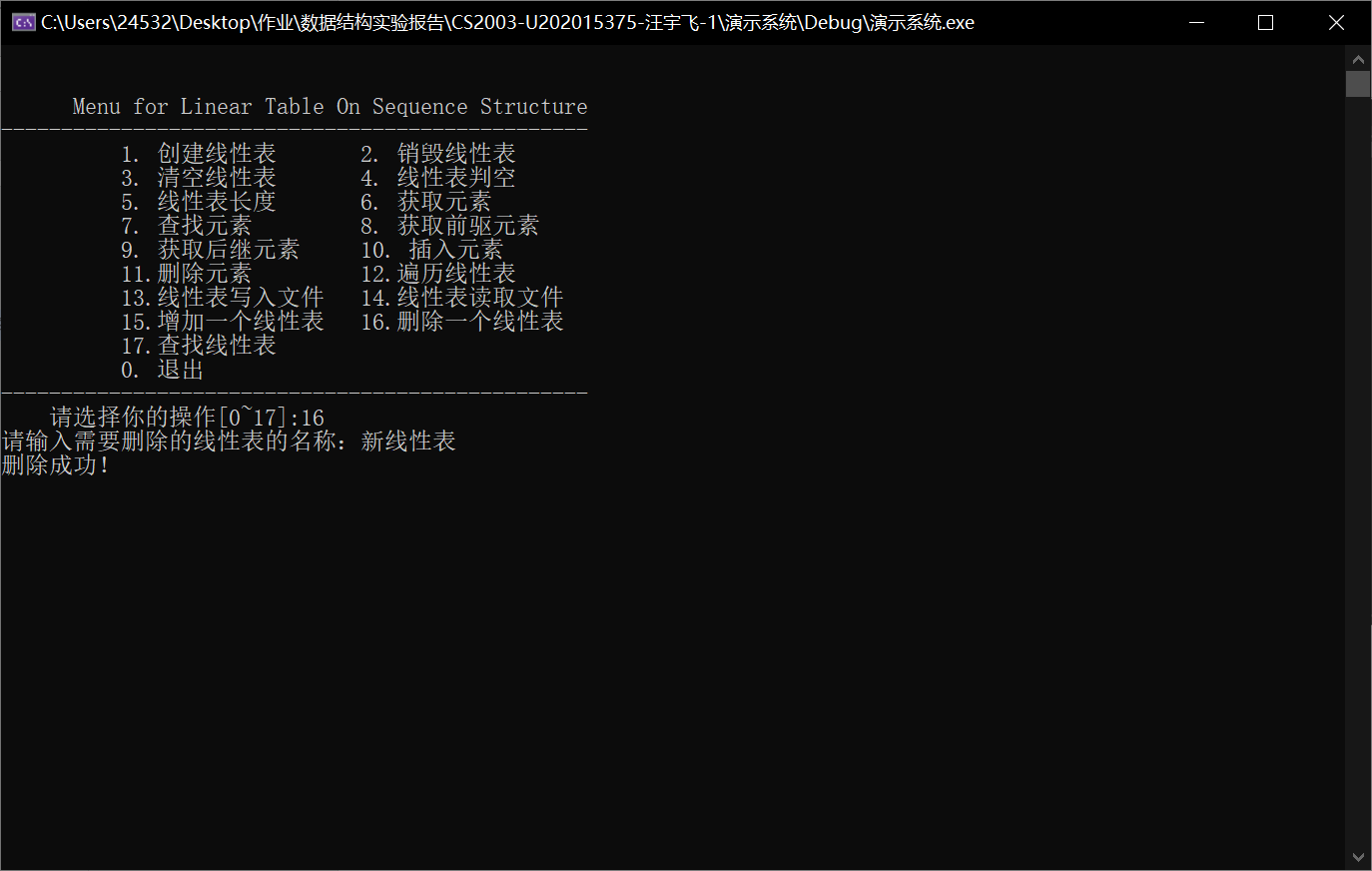


图1-41

用例“不存在的线性表”

测试结果：删除失败（如图1-42）

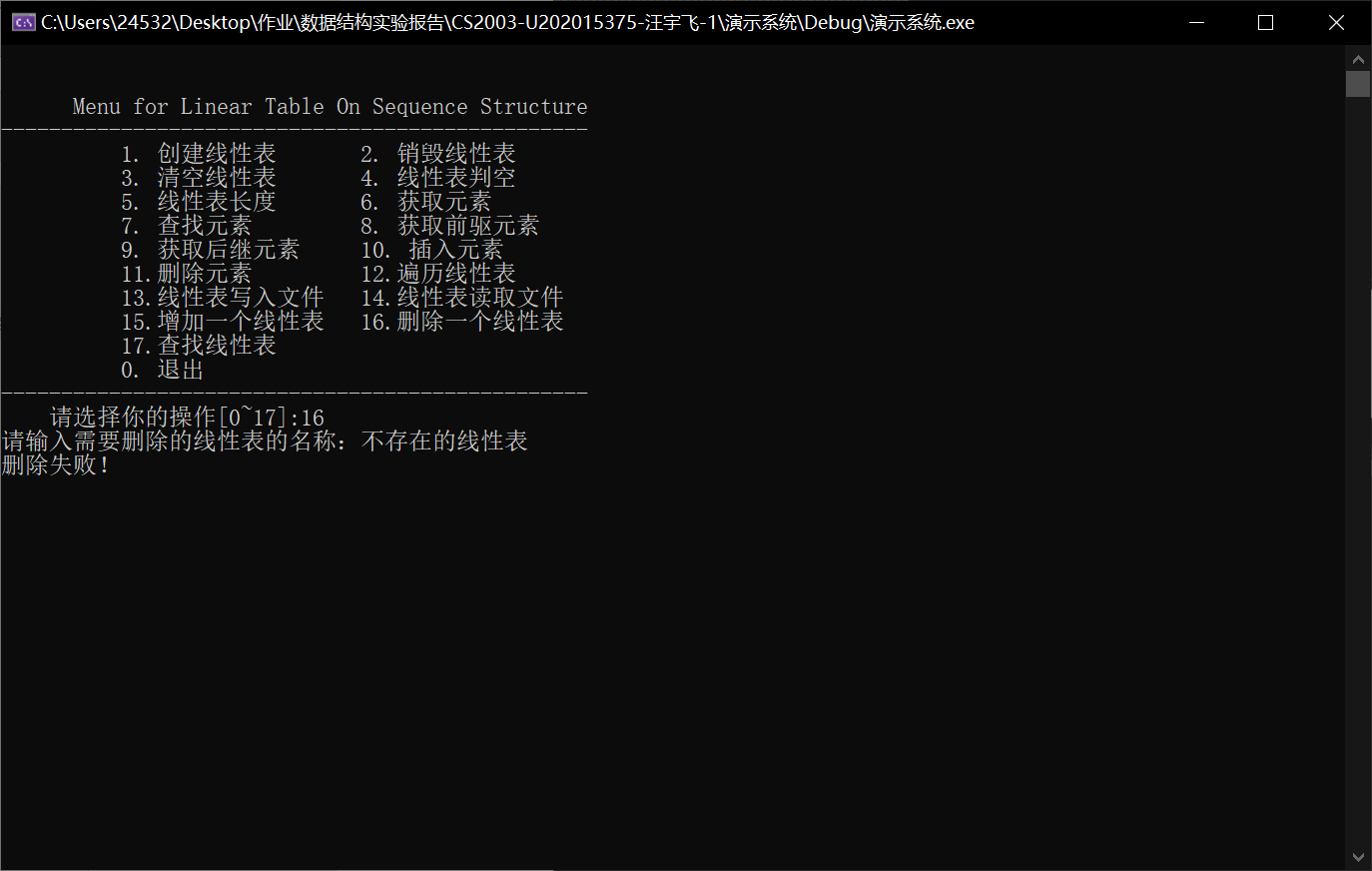


图1-42

1. 查找线性表

用例：线性表集合，其中线性表依次为“第一个线性表”“第二个线性表”“第三个线性表”

测试结果：如图1-43、图1-44、图1-45和图1-46

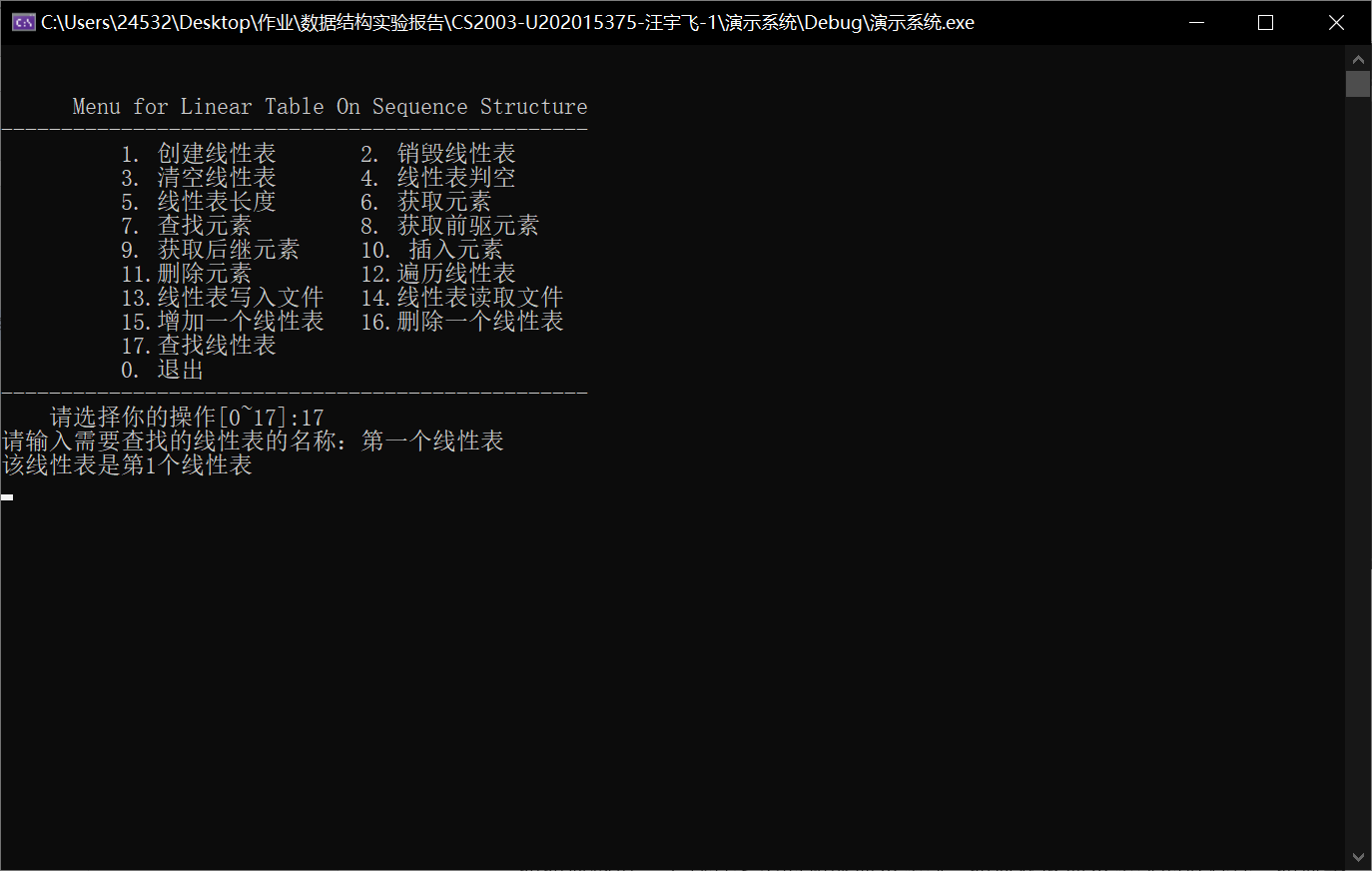


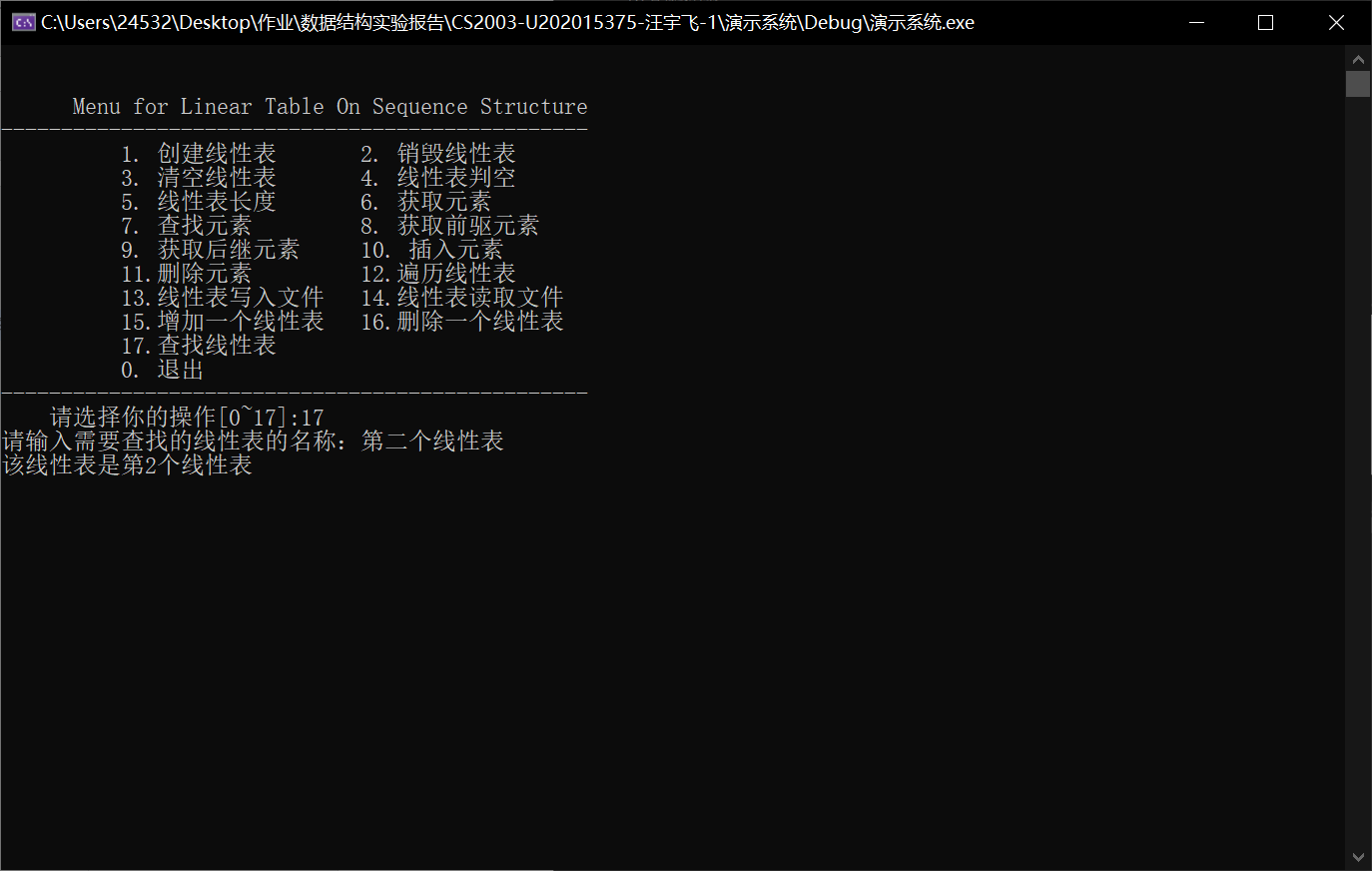
图1-43

图1-44

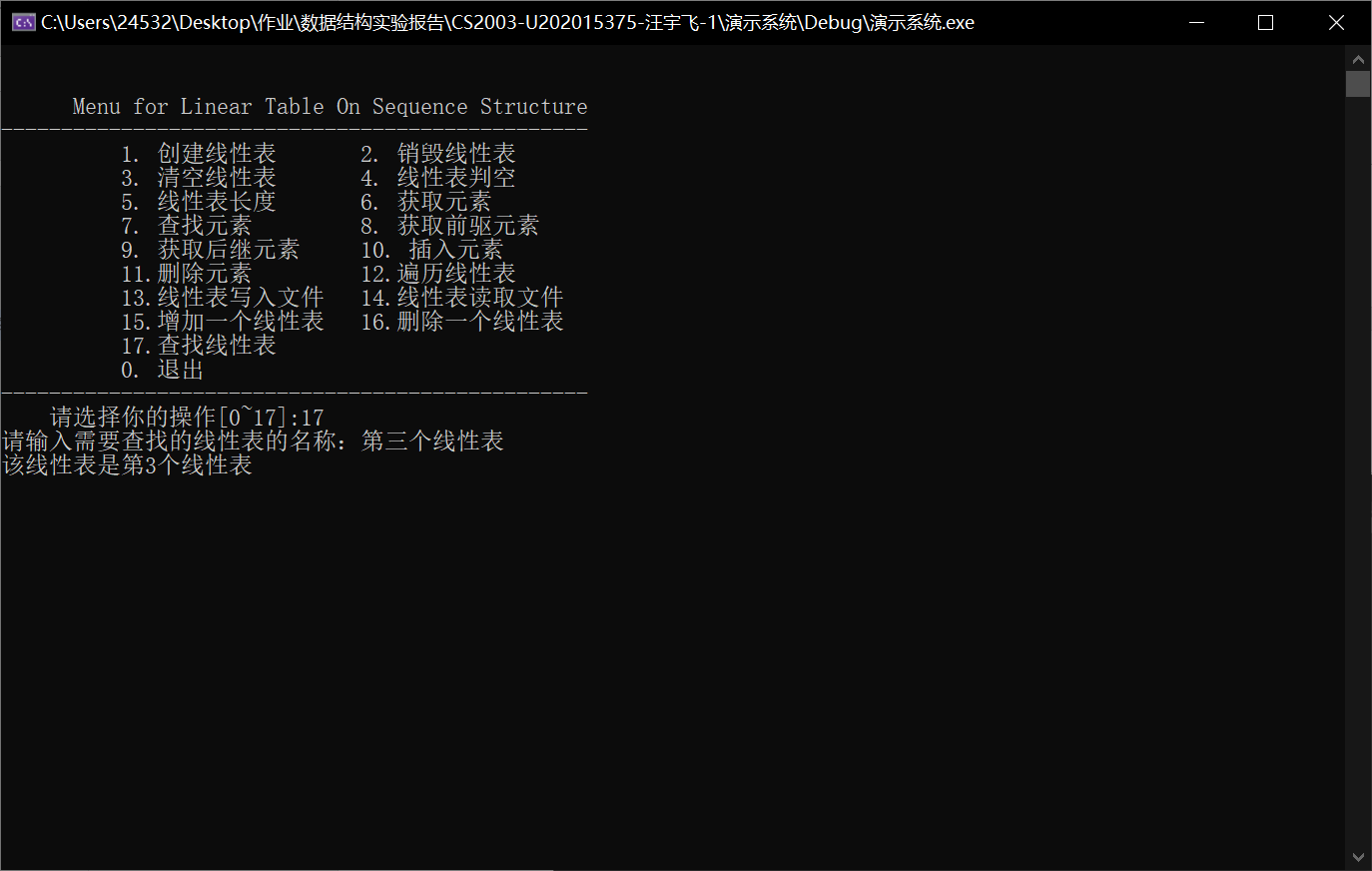


图1-45

图1-46

## 1.5 实验小结

这次是第一次数据结构实验，我对各种相关要求及相应操作都不太熟悉，例如如何构造一个具有菜单的功能演示系统、如何获取演示系统的源程序、如何实现目标程序EXE文件的生成等。

但是通附录中已提供的内容和查找相关查找资料，我学会了使用switch语句来构造演示系统的整体结构、查找C++ source文件获得源程序、对演示系统的代码进行release操作获得独立于IDE运行的EXE文件等操作。

在编写演示系统代码的过程中，我对所使用的函数有了更深的理解，而之前在educoder上完成过关训练时仅仅只知道如何实现函数的功能，但通过编写演示系统，我了解掌握了这些函数的实际应用。

在实现“线性表的读写文件”这一操作中，我复习了文件的相关操作，如fwrite、fread等相关函数的使用。

在实现“多线性表的管理”这一系列操作种，我学会了另一种结构即“线性表集合”的相关定义和操作。