

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： 计算机科学与技术 201801

学 号： U201816030

姓 名： 车春池

指导教师： 周全

报告日期： 2019 年 11 月 7 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.....	2
1.1 问题描述.....	2
1.2 系统设计.....	2
1.3 系统实现.....	2
1.4 实验小结.....	2
2 基于链式存储结构的线性表实现.....	2
2.1 问题描述.....	2
2.2 系统设计.....	2
2.3 系统实现.....	2
2.4 实验小结.....	2
3 基于二叉链表的二叉树实现.....	2
3.1 问题描述.....	2
3.2 系统设计.....	2
3.3 系统实现.....	2

3.4 实验小结.....	2
4 基于二叉链表的二叉树实现.....	2
4.1 问题描述.....	2
4.2 系统设计.....	2
4.3 系统实现.....	2
4.4 实验小结.....	2
参考文献.....	2
附录 A 基于顺序存储结构线性表实现的源程序.....	2
附录 B 基于链式存储结构线性表实现的源程序.....	2
附录 C 基于二叉链表二叉树实现的源程序.....	2
附录 D 基于邻接表图实现的源程序.....	2

1 基于顺序存储结构的线性表实现

1.1 问题描述

依据最小完备性和常用性相结合的原则，以函数形式定义线性表的初始化表、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。采用顺

序表作为线性表的物理结构，实现线性表基本运算。其中 ElemType 为数据元素的类型名，具体含义可自行定义。要求构造一个具有菜单的功能演示系统。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，并给出适当的操作提示显示。演示系统实现线性表的文件形式保存。演示系统实现多个线性表管理。

1.1.1 线性表抽象数据类型

ADT Sqlist{

 数据对象： $D=\{a_i | a_i \text{ 属于 } E\text{lemSet}, i=1, 2, \dots, n, n \geq 0\}$

 数据关系： $R_l=\{<a_{i-1}, a_i> | a_{i-1}, a_i \text{ 属于 } D, i=1, 2, \dots, n\}$

 基本操作：

 InitList(&L)

 初始条件：线性表 L 不存在

 操作结果：构造一个空线性表

 DestroyList (&L)

 初始条件：线性表 L 存在

 操作结果：摧毁线性表 L

ClearList (&L)

初始条件：线性表存在

操作结果：将 L 重置为空表

ListEmpty (&L)

初始条件：线性表 L 存在

操作结果：L 为空表返回 TRUE，否则返回 FALSE

ListLength (&L)

初始条件：线性表 L 存在

操作结果：返回 L 中数据元素的个数

GetElem (L,i,e)

初始条件：线性表 L 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

LocateElem (L,e,compare())

初始条件：线性表存在

操作结果：返回 L 中第 1 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。

PriorElem(L,cur_e,pre_e)

初始条件：线性表存在

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。

$\text{ListInsert}(L, i, e)$

初始条件：线性表存在

操作结果：在 L 的第 i 个位置之前插入新的数据元素 e

$\text{ListDelete}(L, i, e)$

初始条件：线性表存在

操作结果：删除 L 的第 i 个数据元素，用 e 返回其值

$\text{ListTraverse}(L, \text{visit}())$

初始条件：线性表存在

操作结果：依次对 L 的每个数据元素调用函数 $\text{visit}()$

$\text{ListSet}(L)$

初始条件：线性表存在

操作结果：对线性表进行排序

}ADT SqList

1.1.2 多线性表的管理

我定义了一个结构体指针数组，用来存储指向各个线性表的指针值，指针

在数组中对应的下标就是相应线性表的标识（从 0 开始），在演示菜单中可以实现各个线性表的切换。

1.1.3 演示系统和文件存储的设计

演示系统借鉴了 A 中给出的框架，该框架将完成函数调用所需实参值的准备和函数执行结果的实现，并给出适当的操作提示显示，整体以命令行呈现。

此外设计了数据文件实时存储，文件存储的操作包括在演示系统上，用户可以自行选择是否将当前线性表存储，存储方式为将当前的线性表结构体和数据区域直接将内存区块写入文件中。文件读取操作也包括在演示系统里面，读取时用户可以选择当前目录中已经保存的文件加载，来还原该文件所存储的一个线性表的数据。

1.2 系统设计

1.2.1 数据物理结构

1.线性表的存储数据结构

线性表结构体定义如下：

```
//the main struct of linear list
typedef struct
{
    //顺序表（顺序存储）的定义
    ElemtType*elem;//数据元素
    int length;//表当前的长度
```

```
    int listsiz; //表当前容量  
} SqList,*SqListP;
```

2.多线性表的存储数据结构

```
//ListNumber 为管理的线性表个数  
SqListP ListGroup[ListNumber];
```

3.宏定义

```
#define TRUE 1  
  
#define FALSE 0  
  
#define OK 1  
  
#define ERROR 0  
  
#define INFEASTABLE -1  
  
#define OVERFLOW -2  
  
typedef int status;  
  
typedef int ElemType;
```

在本程序中，数据原子类型被定义为整型 int

1.2.2 演示系统

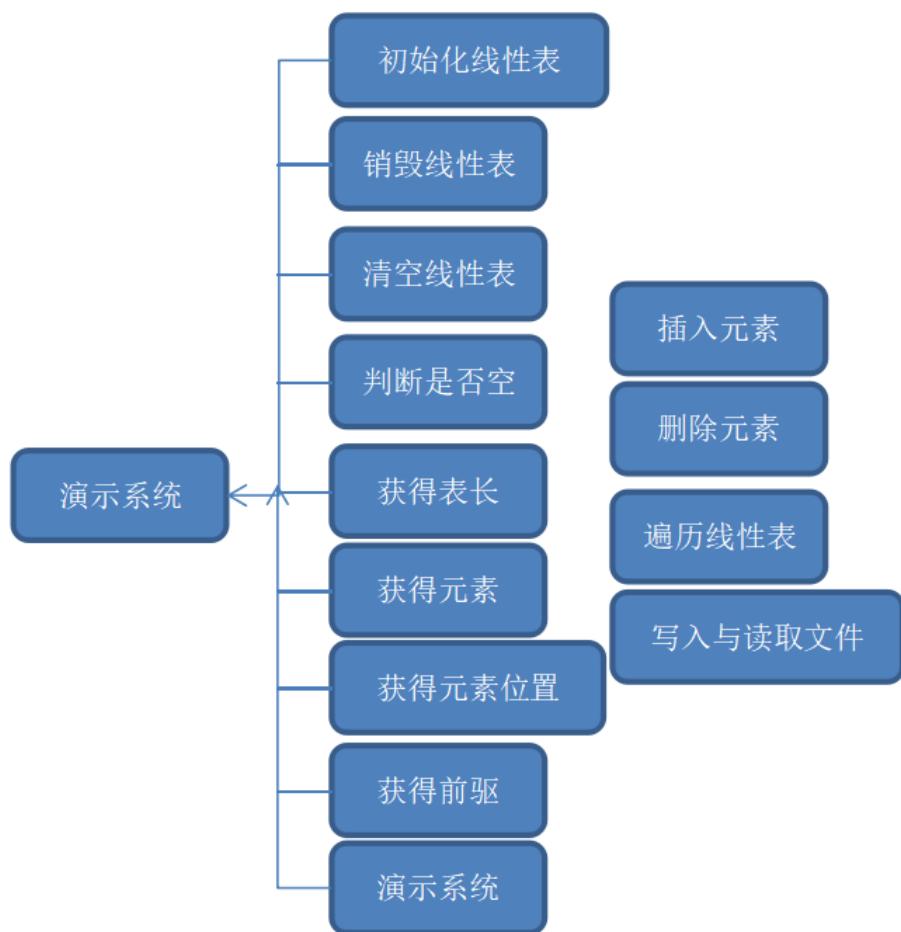
演示系统包括用户操作界面和功能调用部分。

演示系统界面语言为英文，有的操作和提示语言为中文。

用户操作界面输出可选的线性表操作，用户输入数字选择要进行的操作。

在用户选择操作后，系统提示用户输入参数，并且显示参数的信息。

功能调用部分将用户输入的有关信息传递给线性数据结构的操作函数进行调用，并对函数的返回值进行处理判断输出相应的提示信息。



1.2.3 线性表运算实现算法

1. status InitList(SqListP L)

功能：初始化线性表

算法实现：为线性表 L 的 elem 区域申请空间，如果申请失败则返回
ERROR，否则返回 OK。

时空效率分析：算法的时间复杂度为 $O(1)$ ，空间上为 L 的 elem 区
域申请空间，所以空间复杂度为 $O(1)$ 。

2. status DestoryList(SqListP L)

功能：摧毁线性表

算法实现：如果 L 的 elem 区域不为空，则 free 掉 elem 区域，然后将
整个 L 的表长置为 0。

时空效率分析：算法的时间复杂度为 $O(1)$ ，空间复杂度为 $O(1)$

3. status ClearList(SqListP L)

功能：清空线性表

算法实现：如果 L 的 elem 区域不为空，则将 L 的表长置为 0

时空效率分析：算法的时间复杂度为 $O(1)$ ，空间复杂度为 $O(0)$

4. status ListEmpty(SqListP L)

功能：判断线性表是否为空

算法实现：如果 L 的表长为 0，则返回 TRUE，否则返回 FALSE

时空效率分析：时间复杂度为 O (1) ，空间复杂度为 O (1)

5. int ListLength(SqListP L)

功能：求线性表的长度

算法实现：直接返回线性表结构体中存储的 L->length 即可

时空效率分析：时间复杂度为 O (1) ，空间复杂度为 O (1)

6. status GetElem(SqListP L,int i,ElemType* e)

功能：获得线性表中指定位置的数据

算法实现：首先判断指定位置是否合法，若不是则返回 ERROR，否则

把 L->elem[i-1]的值赋给*e，返回 OK

时空效率分析：时间复杂度为 O (1) ，空间复杂度为 O (1)

7. int LocateElem(SqListP L,ElemType e,int(*compare)(ElemType x,ElemType e))

功能：寻找指定元素在线性表中的位置

算法实现：遍历线性表的数据区域，如果当前数据元素与 e 符合 compare () 关系，则，返回 index+1，否则返回 0

时空效率分析：时间复杂度为 O (n) ，空间复杂度为 O (1)

8. status PriorElem(SqListP L,ElemType cur,ElemType&pre_e)

功能：获得指定元素之前的一个一个元素

算法实现：直接遍历寻找，然后把前一位位置的元素的地址赋值给 pre_e。如果找到就返回 OK，否则返回 ERROR。

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

9. status NextElem(SqListP L, ElemType cur, ElemType&next_e)

功能：获得指定元素之后的一个元素

算法实现：直接遍历寻找，然后把后一位位置的元素的地址赋值给 next_e，如果找到就返回 OK，否则返回 ERROR

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

10. status ListInsert(SqListP L, int i, ElemType e)

功能：插入元素

算法实现：首先要判断插入的位置是否合法，如果不合法则返回 ERROR，然后判断 $L->size$ 和 $L->length$ 是否相同，如果相同，则线性表已满，就再给线性表分配固定大小的内存空间，然后将指定位置之后的元素全部向后移动一个位置，将新的元素插入到指定位置，返回 OK

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

11. status ListDelete(SqListP L, int i, ElemType* e)

功能：删除元素

算法实现：首先要判断删除的位置是否合法，若不是则返回 ERROR，

然后将指定位置之后的元素全部向前移动一个位置，返回 OK

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

12. status ListTraverse(SqListP L,void(*visit)(ElemType*e))

功能：遍历线性表并将每个数据元素调用 visit 函数

算法实现：直接遍历并对每个数据元素调用 visit () 函数即可

时空效率分析：时间复杂度 $O(n)$ ，空间复杂度 $O(1)$

13. status ListSort(SqListP L)

功能：对线性表中的元素进行排序

算法实现：运用冒泡排序法对 L 中的元素进行排序

时空效率分析：时间复杂度 $O(n^2)$ ，空间复杂度 $O(1)$

1.2.4 多线性表管理实现算法

由于多线性表管理采用结构体指针数组管理，只涉及到查找操作，而且查找操作直接利用数组下标，所以时间复杂度为 $O(1)$ ，空间复杂度为 $O(1)$

1.2.5 文件存储实现算法

1. 写文件

算法实现：用户输入要保存的文件名，打开文件，根据线性表的 $L->size$ 存入数据空间，将当前的线性表作为文件保存，数据保存完毕，关闭文件。

时空效率分析：时间复杂度为 $O(1)$ ，空间复杂度为 $O(n)$

2. 读取文件

算法实现：用户输入要读取的文件名，打开文件，读取该文件的数据存入当前线性表，直到文件中的数据读入完成，关闭文件。

时空效率分析：时间复杂度为 $O(1)$ ，空间复杂度为 (n)

1.3 系统实现

1.3.1 实验环境

实验环境为 Windows10，编译器版本为 TDM-GCC 4.7.1，代码采用开源的编译器 Dev-C++ 编写。由指定的 MakeFile 来完成编译。

文件说明：

*LinearList.h：线性表库头文件

*test.cpp：线性表及演示系统实现

*LinearList_1.dev：DEV 文件

*LinearList_1.exe：可执行文件

*LinearList_1.layout：LAYOUT 文件

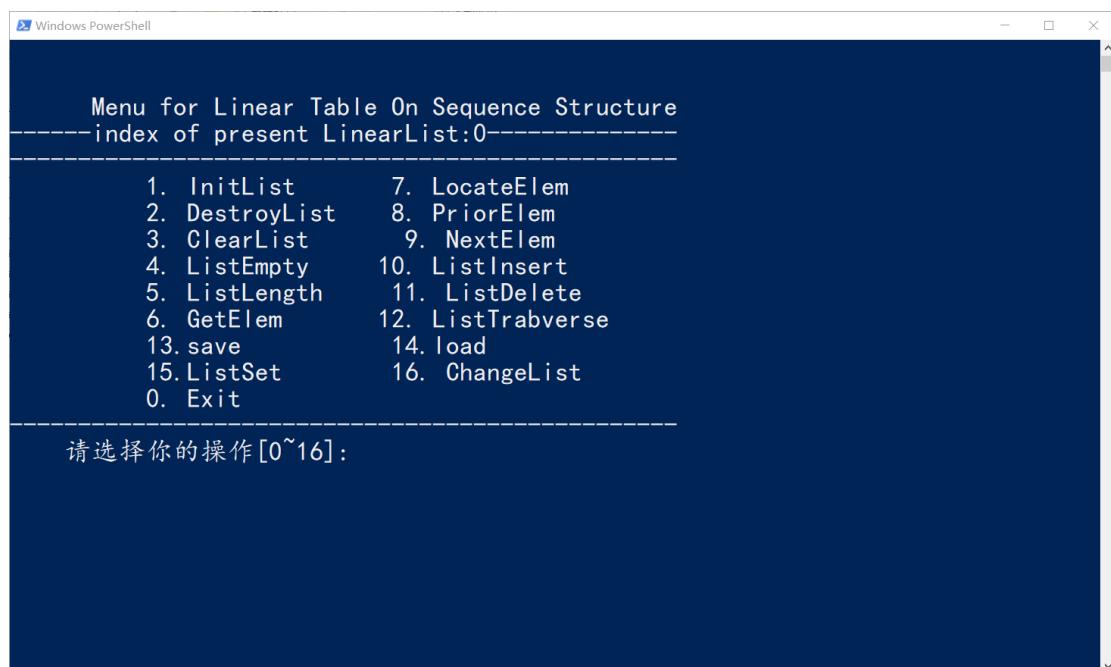
*Makefile.win:自动化编译命令

1.3.2 代码亮点

所有代码书写借鉴 Google C/C++标准代码规范，函数库在必要的地方加上了注释，符合生产规范。错误检查和提示全面，根据不同的错误信息会有不同的提示信息。

1.3.3 操作演示

界面演示：



```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16] :
```

图 1-2 演示系统界面

1. 首先创建一个线性表

```
Windows PowerShell
----- Menu for Linear Table On Sequence Structure ----- index of present LinearList:0-----
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. save       14. load
15. ListSet    16. ChangeList
0. Exit

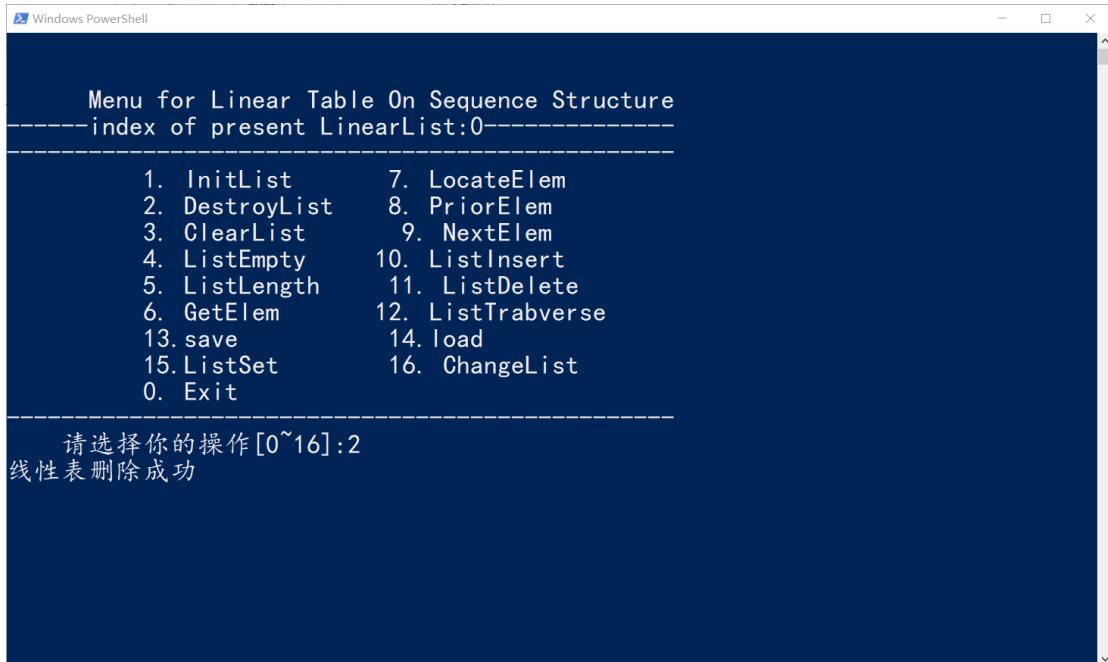
请选择你的操作[0~16]:1
线性表创建成功!
```

2. 然后用 ListEmpty 查看，显示这个新创建的表是空表

```
Windows PowerShell
----- Menu for Linear Table On Sequence Structure ----- index of present LinearList:0-----
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. save       14. load
15. ListSet    16. ChangeList
0. Exit

请选择你的操作[0~16]:4
空表
```

3. 然后删除这个表

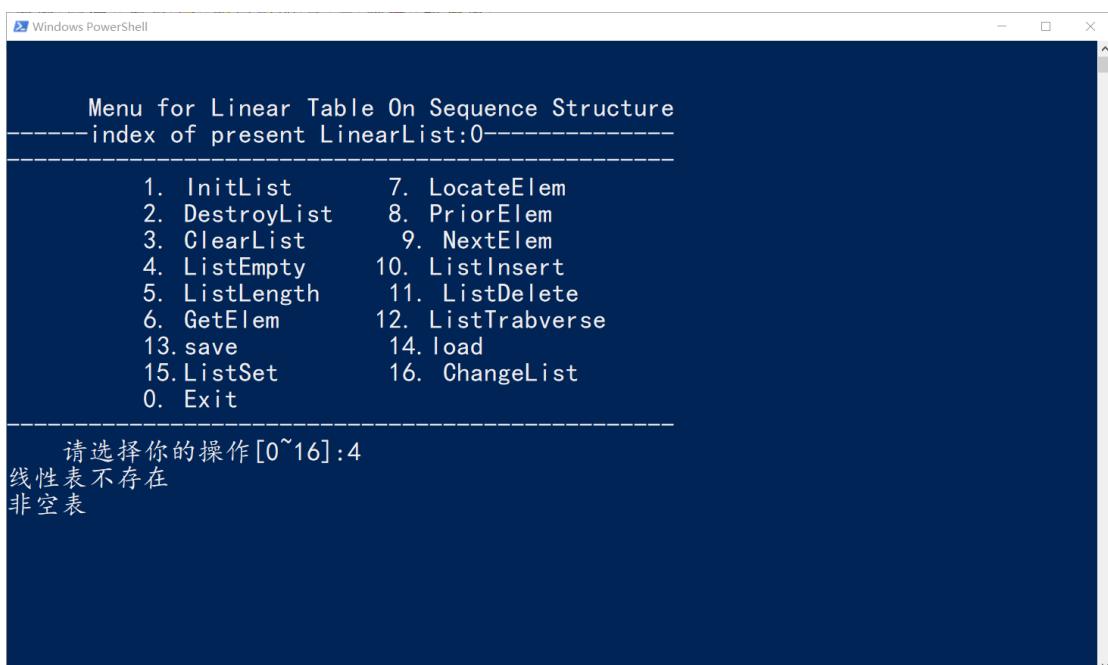


```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:2
线性表删除成功
```

4. 再用 ListEmpty 查看，显示线性表不存在



```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:4
线性表不存在
非空表
```

5. 重新创建一个线性表

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:1
线性表创建成功!
```

6. 依次按顺序插入三个元素 1,2,3

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:10
输入i, i为插入元素的位序: 1
输入e, e为插入元素的值: 1
插入成功
```

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:0

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:10
输入i, i为插入元素的位序: 2
输入e, e为插入元素的值: 2
插入成功
```

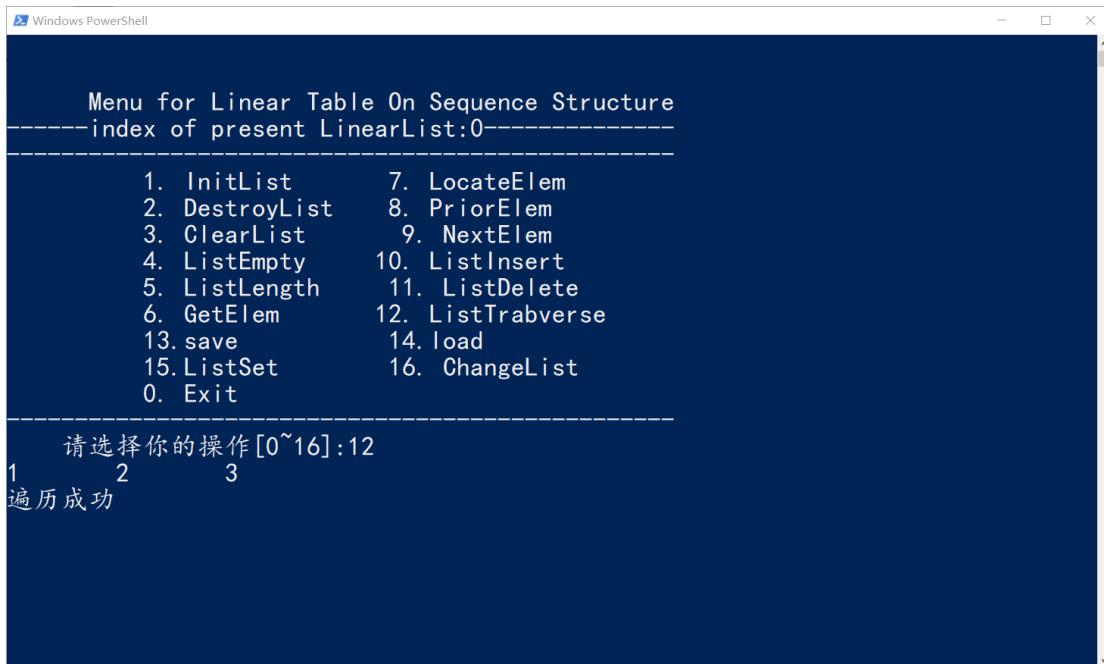
```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:0

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:10
输入i, i为插入元素的位序: 3
输入e, e为插入元素的值: 3
插入成功
```

7. 打印这个表（操作 12 中的 visit (x) 函数是将 x 打印出来）

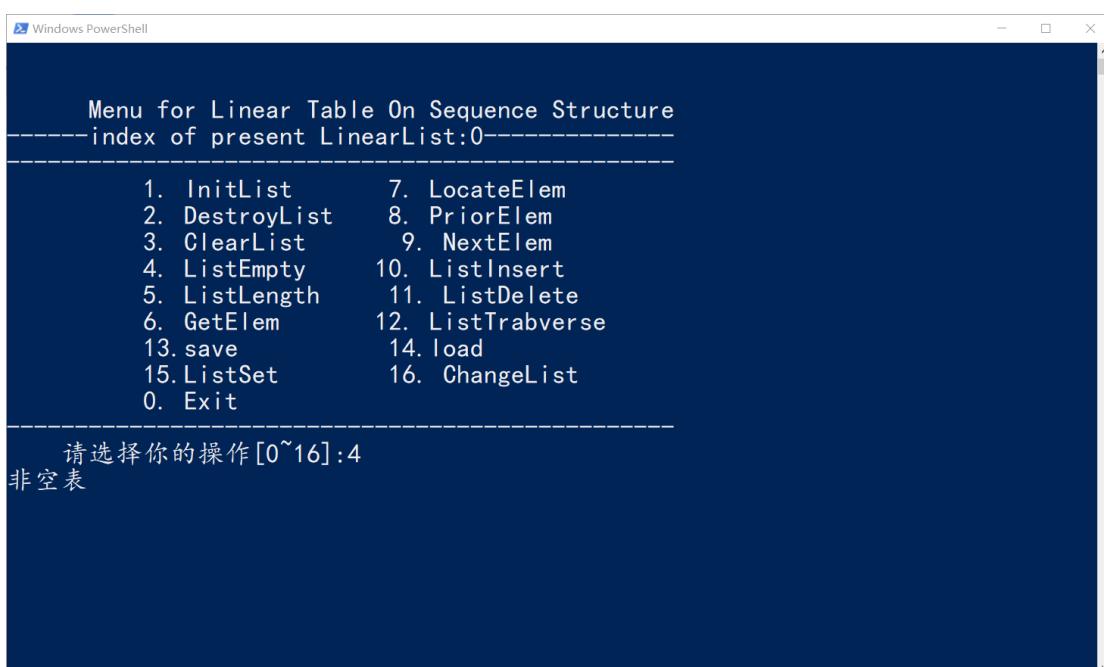


```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:12
1      2      3
遍历成功
```

8. 用 ListEmpty 查看，显示这是个非空表

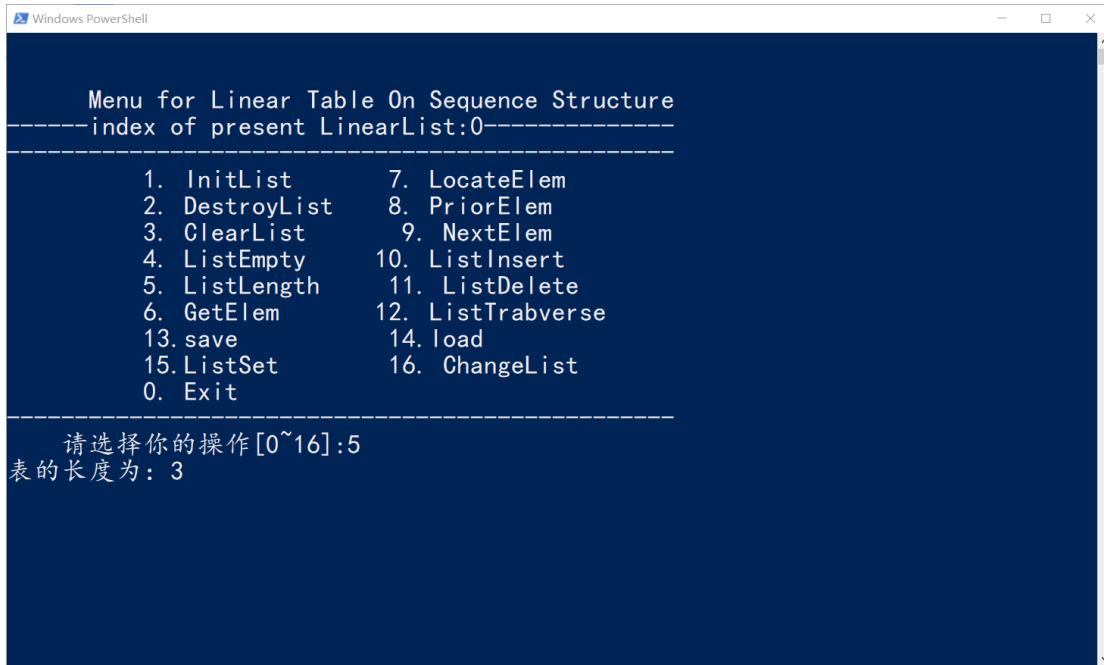


```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:4
非空表
```

9. 查看这个表的长度，显示是 3

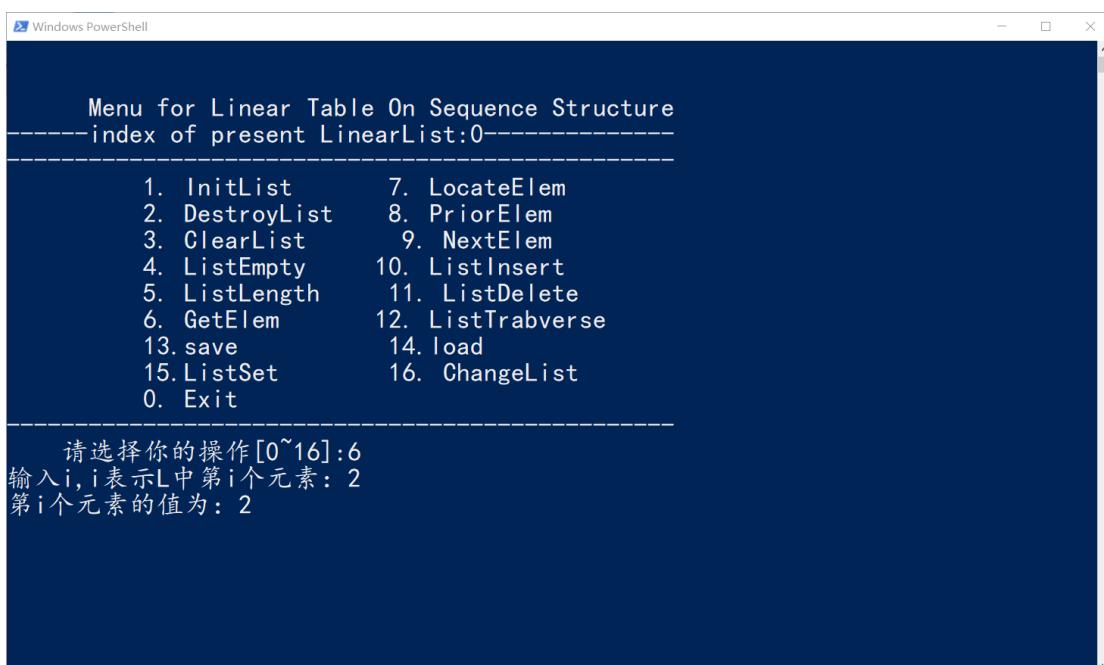


```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:5
表的长度为: 3
```

10. 获得这个表中第 2 个数据元素，显示是 2



```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:6
输入i, i表示L中第i个元素: 2
第i个元素的值为: 2
```

11. 获得这个表中第一个与 3 相等的数据元素（操作 7 的 compare (x, y)

函数是判断 x 与 y 是否相等）

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:7
输入e, e为目标值: 3
第一个满足条件的位序为: 3
```

12.获得元素 2 之前的元素

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:8
输入cur, 下面将获得cur的前驱 : 2
前驱为: 1
```

13.获得元素 2 之后的元素

Windows PowerShell

```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:9
输入cur, 下面将获得cur的后驱: 2
后驱为: 3
```

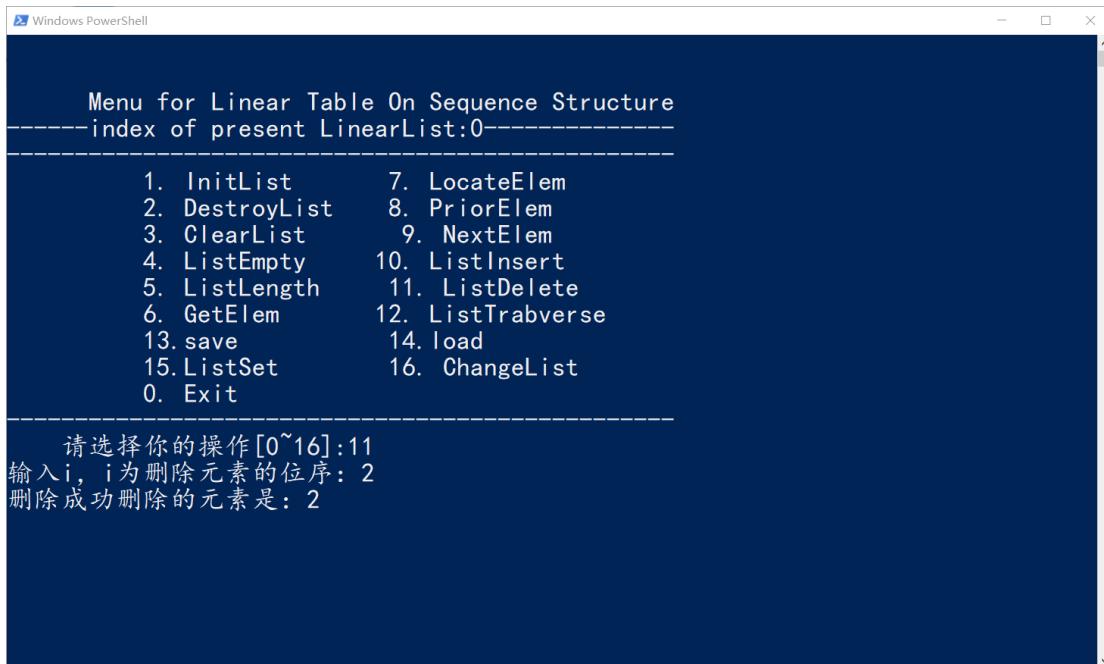
14. 将当前线性表的数据写入文件“**data_01**”

Windows PowerShell

```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:13
input_file_name:data_01
文件写入成功
```

15. 删除当前表的第 2 个元素

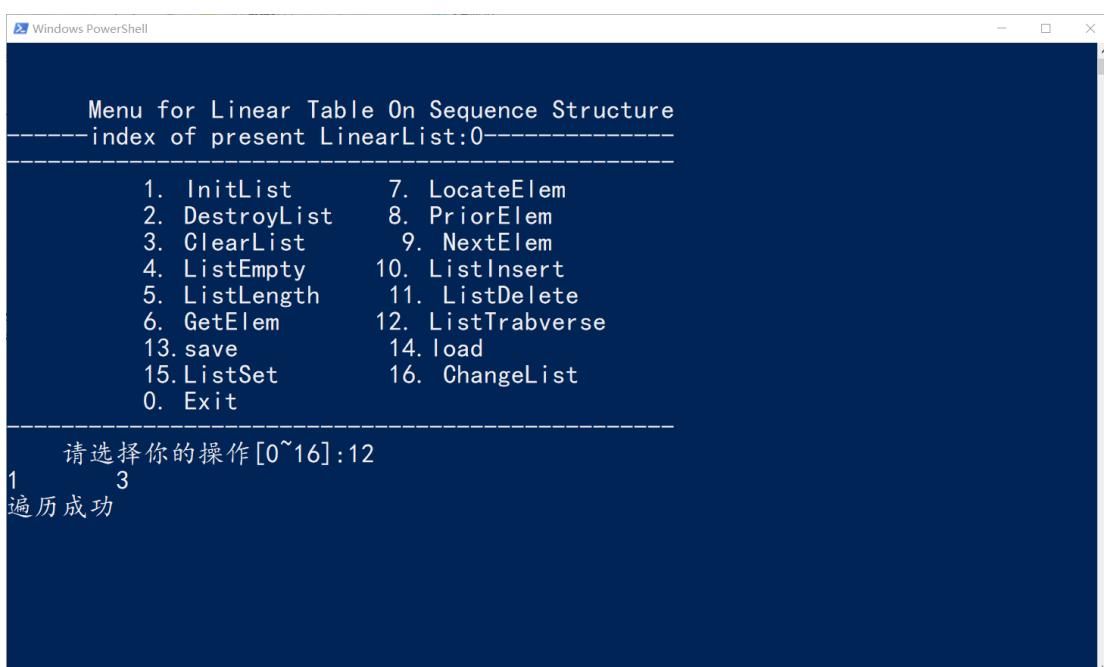


```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:11
输入i, i为删除元素的位序: 2
删除成功删除的元素是: 2
```

16. 打印这个表，发现第 2 个元素成功被删除



```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:12
1      3
遍历成功
```

17. 在表的第 3 个位置插入值 5

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:10
输入i, i为插入元素的位序: 3
输入e, e为插入元素的值: 5
插入成功
```

18. 打印这个表，发现插入数据成功

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:12
1      3      5
遍历成功
```

19. 再将当前表的数据写入文件 “data_02”

```
Windows PowerShell
-----[index of present LinearList:0]-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:13
input file name:data_02
文件写入成功
```

20.切换线性表，切换索引为 1

```
Windows PowerShell
-----[index of present LinearList:0]-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:16
输入想要切换的线性表的索引1
线性表不存在, 请用InitList函数创建
线性表切换成功
```

21.创建一个新表

Windows PowerShell

```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:1
线性表创建成功!
```

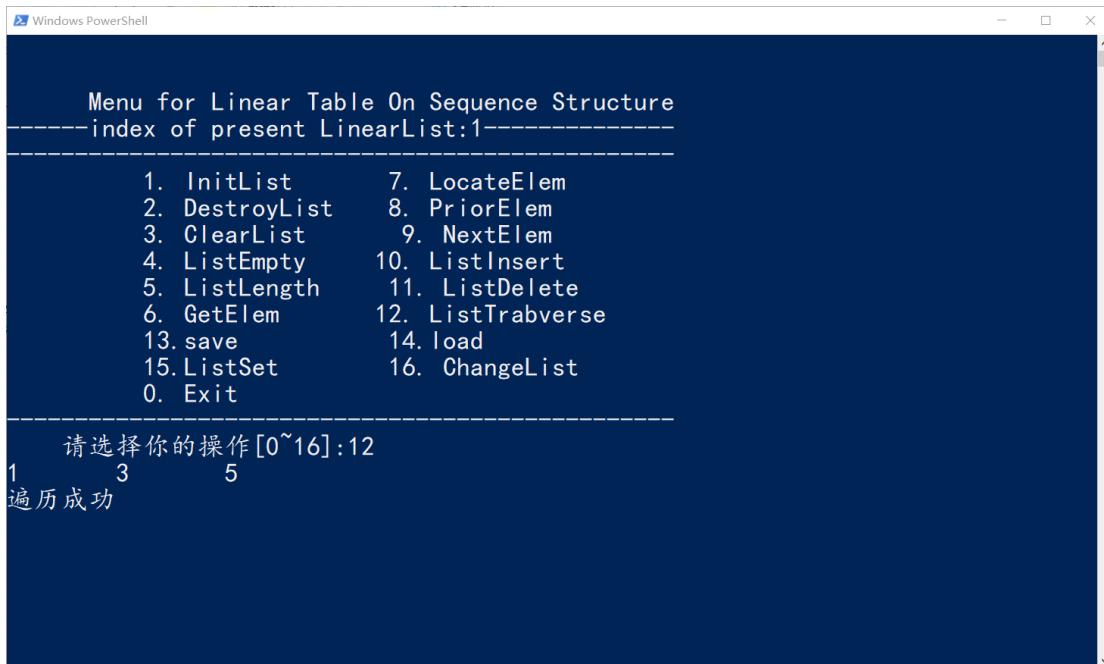
22. 将文件“data_02”的数据加载进当前线性表

Windows PowerShell

```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:14
input_file_name:data_02
文件读取成功
```

23. 打印这个表，发现数据被成功加载

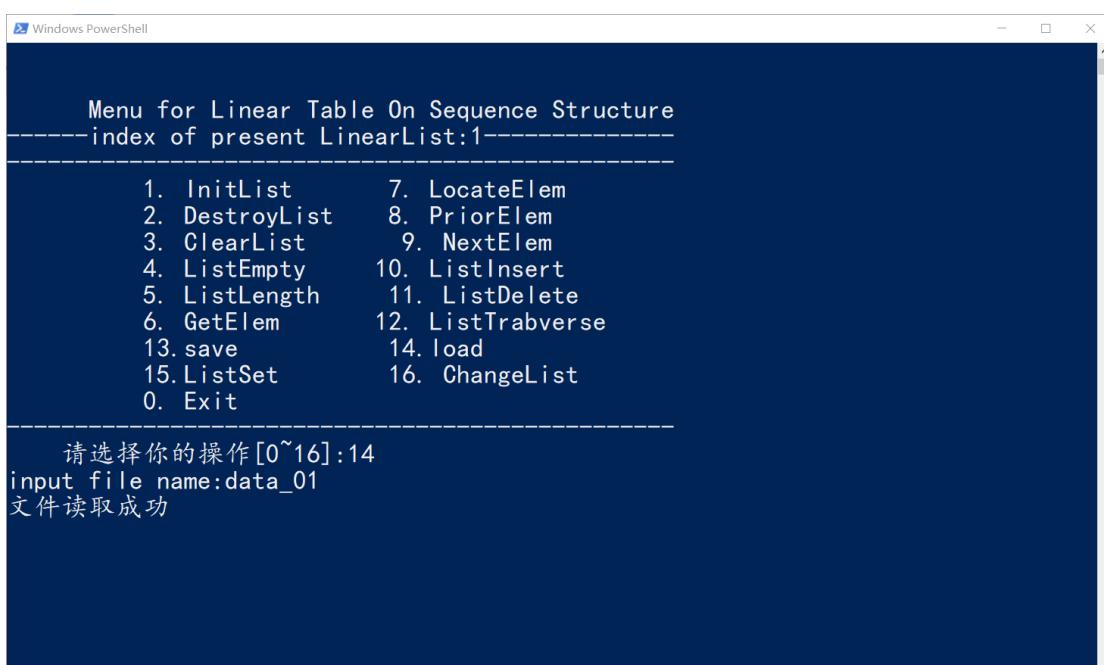


```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:12
1      3      5
遍历成功
```

24.再将文件“data_01”的数据加载进当前线性表



```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:14
input_file_name:data_01
文件读取成功
```

25.打印这个表，发现数据也被成功加载

Windows PowerShell

```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:12
1      2      3
遍历成功
```

26. 切换回索引为 0 的线性表

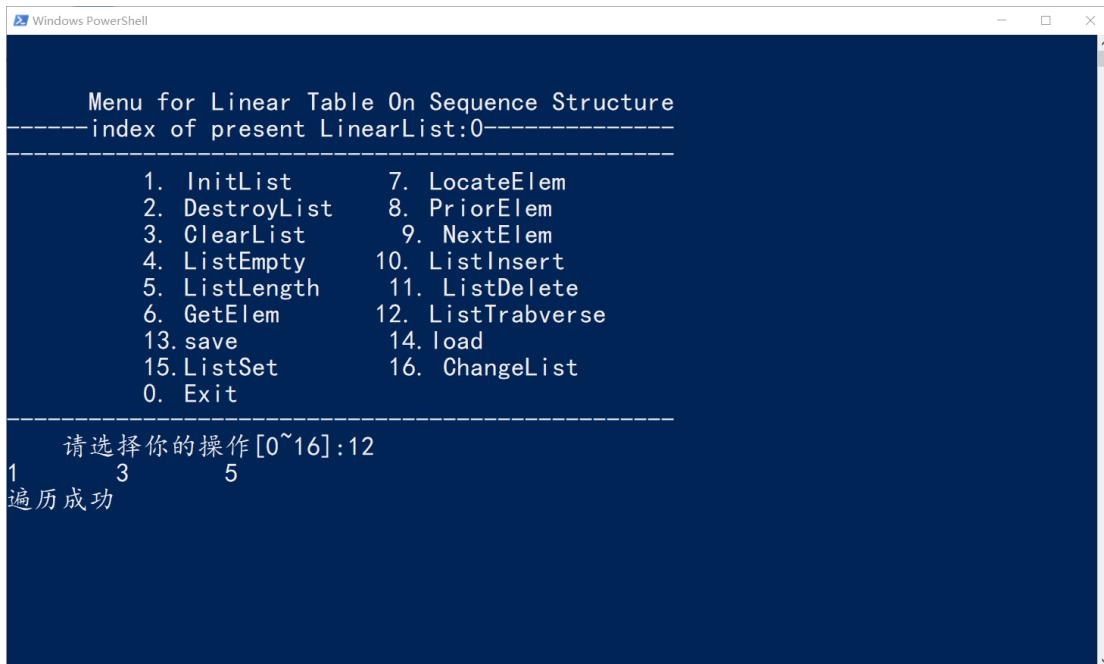
Windows PowerShell

```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:16
输入想要切换的线性表的索引0
线性表切换成功
```

27. 打印当前表，显示的数据与 `data_02` 中的数据相同，说明各线性表之间的操作相互独立

操作相互独立

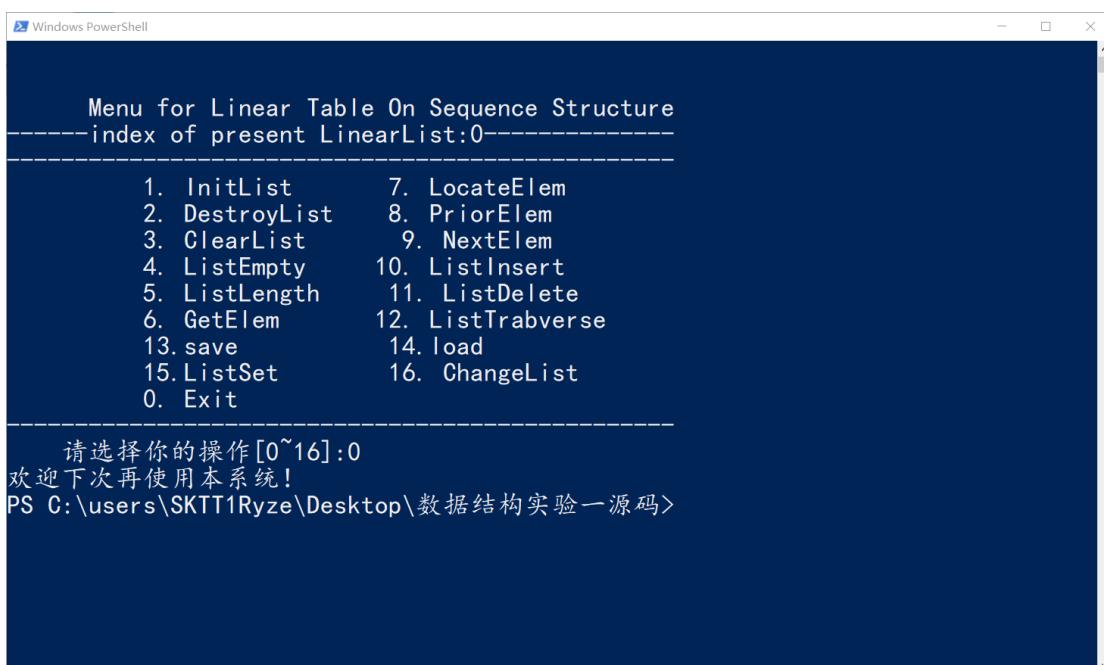


```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:12
1      3      5
遍历成功
```

28.正常退出系统



```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:0
欢迎下次再使用本系统!
PS C:\users\SKTT1Ryze\Desktop\数据结构实验一源码>
```

通过以上的演示证明，我们的线性表的功能是完整实现了的，可以演示要求的十二种线性表上的运算，并且做到了多线性表管理，实现了文件存储功能，不

存在内存泄漏，完整的达到了实验要求。

1.3.3 系统测试

在上面操作的基础上，我们再创建两个线性表，索引分别为 2 和 3，索引为 2 的线性表中的数据元素是 1,4,2,7,5；索引为 3 的线性表中的数据元素是 12,4,10,3,6；分别写入文件“data_03”和“data_04”

下面以这两个线性表基础，进行系统测试。

1. 初始化已存在的线性表

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:2-----
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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:1
线性表存在
线性表创建失败!
```

2. 删除不存在的线性表

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:2
线性表不存在
线性表删除失败!
```

3. 清空不存在的线性表

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:3
线性表不存在
失败
```

4. 判断不存在的线性表是否为空

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:4
线性表不存在
非空表
```

5. 获得不存在的线性表的表长

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:5
线性表不存在
表的长度为: 0
```

6. 获得表中不存在的元素

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:6
输入i, i表示L中第i个元素: 10
OVERFLOW失败
```

7. 确定表中不存在的元素的位置

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:7
输入e, e为目标值: 20
满足条件的数据元素不存在
第一个满足条件的位序为: 0
```

8. 获得表中不存在的元素的前元素

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:8
输入cur, 下面将获得cur的前驱 : 20
cur不在数据元素里
失败
```

9. 获得表中第一个元素的前元素

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:8
输入cur, 下面将获得cur的前驱 : 1
cur是第一个
失败
```

10. 获得表中不存在的元素的后元素

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:9
输入cur, 下面将获得cur的后驱: 20
cur不在数据元素里
失败
```

11. 获得表中最后一个元素的后元素

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:9
输入cur, 下面将获得cur的后驱: 5
cur是最后一个
失败
```

12. 插入元素超过线性表的最大大小

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:10
输入i, i为插入元素的位序: 10
输入e, e为插入元素的值: 10
OVERFLOW失败
```

13. 删除表中不存在的元素

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
index of present LinearList:2

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. save         14. load
15. ListSet      16. ChangeList
0. Exit

请选择你的操作[0~16]:11
输入i, i为删除元素的位序: 10
OVERFLOW失败
```

14. 打印不存在的线性表

```
Windows PowerShell
-----  
Menu for Linear Table On Sequence Structure  
-----  
index of present LinearList:2-----  
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. save         14. load  
15. ListSet      16. ChangeList  
0. Exit  
-----  
请选择你的操作[0~16]:12  
线性表不存在  
失败
```

1.4 实验小结

这次实验是数据结构实验课程的第一次实验，难度不算特别大，但还是要注意一些细节，比如必要的注释，错误输入提醒，系统健壮性的实现等等，个人感觉这次实验还有许多做得不足之处，希望后面几次实验能将这些不足改进，达到提高我的编程能力的目的。

2 基于链式存储结构的线性表实现

2.1 问题描述

实现基于链式存储结构的线性表，并实现线性表的基本运算。

构造一个具有菜单的功能演示系统。

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。

演示系统实现了线性表的文件形式存储。

演示系统可以实现多线性表管理。

整个系统的设计模式如下：

2.1.1 线性表抽象数据类型

依据最小完备性和常用性相结合的原则，设计了线性表的数据对象和数据关系，并定义了线性表的初始化表，销毁表，清空表，判定空表，求表长和和获得元素等 12 种基本运算，具体数据和运算功能定义如下。

ADT Sqlist{

 数据对象： $D = \{a_i | a_i \text{ 属于 } ElemSet, i=1, 2, \dots, n, n \geq 0\}$

 数据关系： $R_l = \{<a_{i-1}, a_i> | a_{i-1}, a_i \text{ 属于 } D, i=1, 2, \dots, n\}$

 基本操作：

 InitList(&L)

 初始条件：线性表 L 不存在

操作结果：构造一个空线性表

DestroyList (&L)

初始条件：线性表 L 存在

操作结果：摧毁线性表 L

ClearList (&L)

初始条件：线性表存在

操作结果：将 L 重置为空表

ListEmpty (&L)

初始条件：线性表 L 存在

操作结果：L 为空表返回 TRUE，否则返回 FALSE

ListLength (&L)

初始条件：线性表 L 存在

操作结果：返回 L 中数据元素的个数

GetElem (L,i,e)

初始条件：线性表 L 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

LocateElem (L,e,compare())

初始条件：线性表存在

操作结果：返回 L 中第 1 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。

PriorElem(L,cur_e,pre_e)

初始条件：线性表存在

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。

ListInsert(L,i,e)

初始条件：线性表存在

操作结果：在 L 的第 i 个位置之前插入新的数据元素 e

ListDelete(L,i,e)

初始条件：线性表存在

操作结果：删除 L 的第 i 个数据元素，用 e 返回其值

ListTraverse(L,visit())

初始条件：线性表存在

操作结果：依次对 L 的每个数据元素调用函数 visit()

ListSet(L)

初始条件：线性表存在

操作结果：对线性表进行排序

```
}ADT SqList
```

2.1.2 多线性表的管理

同实验一一样我定义了一个结构体指针数组，用来存储指向各个线性表的指针值，指针在数组中对应的下标就是相应线性表的标识（从 0 开始），在演示菜单中可以实现各个线性表的切换。

2.1.3 演示系统和文件存储的设计

演示系统借鉴了附录 A 中给出的框架，该框架将完成函数调用所需实参值的准备和函数执行结果的实现，并给出适当的操作提示显示，整体以命令行呈现。

此外设计了数据文件实时存储，文件存储的操作包括在演示系统上，用户可以自行选择是否将当前线性表存储，存储方式为将当前的线性表结构体和数据区域直接将内存区块写入文件中。文件读取操作也包括在演示系统里面，读取时用户可以选择当前目录中已经保存的文件加载，来还原该文件所存储的一个线性表的数据。

2.2 系统设计

2.2.1 数据物理结构

线性表的存储数据结构

线性表结构体定义如下：

```
typedef struct LNode//define of ADT
{
    ElemType data;//data area
    struct LNode *next;//pointer area
}LNode,*LinkList;
```

宏定义：

```
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASTABLE -1
#define OVERFLOW -2
typedef int status;
typedef int ElemType;
```

2.2.2 演示系统

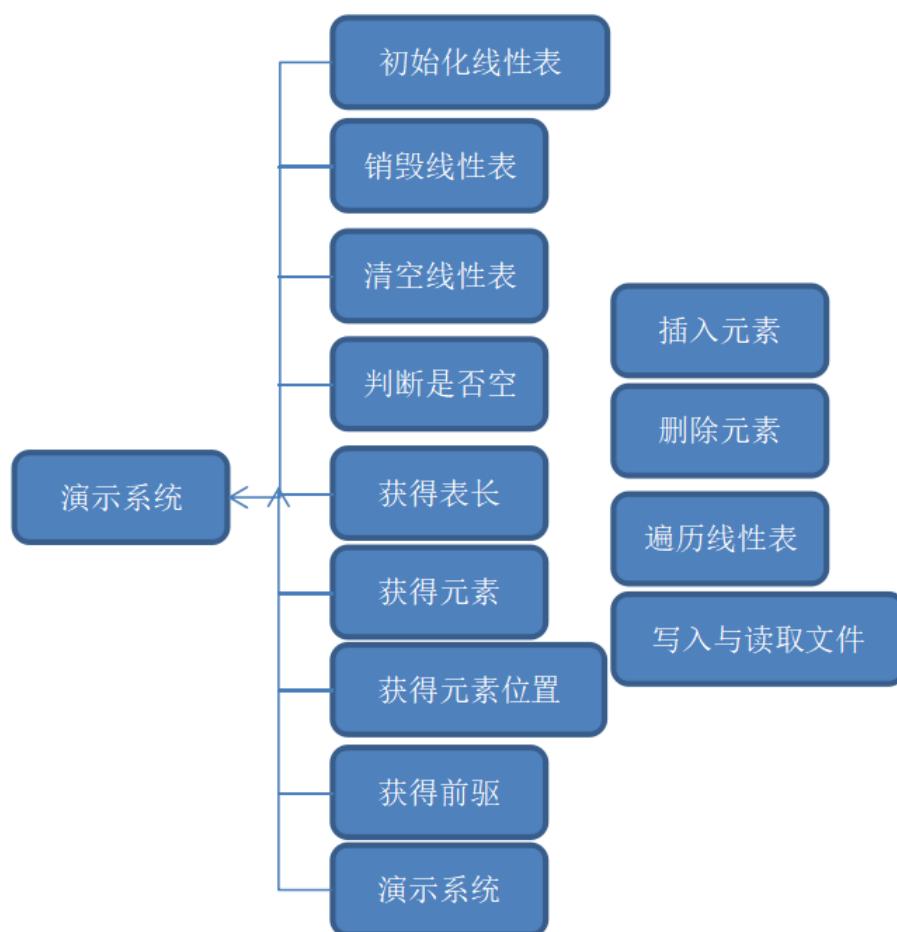
演示系统包括用户操作界面和功能调用部分。

演示系统界面语言为英文，所有操作和提示语言均为英文。

用户操作界面输出可选的线性表操作，用户输入数字选择要进行的操作。

在用户选择操作后，功能调用部分会显示函数的名称，参数，返回值和作用，
系统提示用户输入参数。

功能调用部分将用户输入的有关信息传递给线性数据结构的操作函数进行
调用，并对函数的返回值进行处理判断输出相应的提示信息。



2.2.3 线性表运算实现算法

1. status InitList(LinkList&L);

功能：初始化线性表

算法实现：如果 L 不为空，则打印 L 已经存在，返回 ERROR；否则为 L 申请内存空间，并将节点数据初始化，返回 OK

时空效率分析：算法的时间复杂度为 $O(1)$ ，空间上为 L 指向的节点申请空间，所以空间复杂度为 $O(1)$ 。

2. status DestroyList(LinkList&L);

功能：摧毁线性表

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；如果 L 不为空，则 free 整个链表的每个节点的存储空间，然后让 L 指向 NULL，防止 L 称为野指针，返回 OK。

时空效率分析：算法的时间复杂度为 $O(1)$ ，空间复杂度为 $O(1)$

3. Status ClearList(LinkList&L);

功能：清空线性表

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；如果 L 不为空，则将 L 头结点后面的节点 free 掉，然后将表长置为零

时空效率分析：算法的时间复杂度为 $O(1)$ ，空间复杂度为 $O(0)$

4. Status ListEmpty(LinkList&L);

功能：判断线性表是否为空

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；如果 L 不存在，打印错误信息，返回 ERROR，否则判断表长是否为 0，为 0 返回 TRUE，部位 0 返回 FALSE。

时空效率分析：时间复杂度为 O (1) ，空间复杂度为 O (1)

5. Int ListLength(LinkList&L);

功能：求线性表的长度

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；直接返回线性表头结点的 L->data (表示表长) 即可

时空效率分析：时间复杂度为 O (1) ，空间复杂度为 O (1)

6. Status GetElem(LinkList&L,int i,ElemType&e);

功能：获得线性表中指定位置的数据

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；首先判断指定位置是否合法，若不是则返回 ERROR，否则遍历链表，把链表中第 i 个节点的 data 值赋给 e，返回 OK

时空效率分析：时间复杂度为 O (1) ，空间复杂度为 O (1)

7. Int LocateElem(LinkList&L,ElemType e,int(*compare)(ElemType x,ElemType e));

功能：寻找指定元素在线性表中的位置

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；遍历链表，

直到找到与 e 符合关系 compare 的值，返回给节点的位序，返回 OK，如果没找到，则打印错误信息，返回 ERROR

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

8. Status PriorElem(LinkList&L,ElemType cur,ElemType&pre_e);

功能：获得指定元素之前的一个一个元素

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；直接遍历寻找，然后把前一位位置的节点中的 data 成员的地址赋值给 pre_e。如果找到就返回 OK，否则返回 ERROR。

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

9. Status NextrElem(LinkList&L,ElemType cur,ElemType&next_e);

功能：获得指定元素之后的一个元素

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；直接遍历寻找，然后把后一位位置的节点中的 data 成员的地址赋值给 next_e，如果找到就返回 OK，否则返回 ERROR

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

10. Status ListInsert(LinkList&L,int i,ElemType e);

功能：插入元素

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；首先要判断插入的位置是否合法，如果不合法则返回 ERROR，然后找到要插入位置的

前一个节点，在插入位置新建一个节点，将该节点 data 成员的值赋值为 e，返回 OK

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

11. Status ListDelete(LinkList&L,int i,EemType&e);

功能：删除元素

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；首先要判断删除的位置是否合法，若不是则返回 ERROR，然后将指定位置的节点 free 掉，free 掉之后恢复链表的链接，返回 OK

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(1)$

12. Status ListTraverse(LinkList&L,void(*visit)(ElemType e));

功能：遍历线性表并将每个数据元素调用 visit 函数

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；直接遍历链表对每个节点的 data 成员调用 visit 函数即可

时空效率分析：时间复杂度 $O(n)$ ，空间复杂度 $O(1)$

13. Status ListSort(LinkList&L);

功能：对线性表中的元素进行排序

算法实现：如果 L 为空，则打印错误信息，返回 ERROR；运用冒泡排序法对链表中的节点进行排序

时空效率分析：时间复杂度 $O(n^2)$ ，空间复杂度 $O(1)$

14. ListMerger(LinkList&La,LinkList&Lb,LinkList&Lc);

功能：合并两个线性表

算法实现：如果 La 或 Lb 或 Lc 为空，则打印错误信息，返回 ERROR；先为 Lc 头节点创建长度为 $\text{len}(\text{La}) + \text{len}(\text{Lb})$ 的链表，并将 La 和 Lb 中每个节点的值复制到 Lc 中，然后对 Lc 进行排序，排序成功，返回 OK，否则返回 ERROR

时空效率分析：时间复杂度 $O(n^2)$ ，空间复杂度： $O(n)$

2.2.4 多线性表管理实现算法

由于多线性表管理采用结构体指针数组管理，只涉及到查找操作，而且查找操作直接利用数组下标，所以时间复杂度为 $O(1)$ ，空间复杂度为 $O(1)$

2.2.5 文件存储实现算法

1. 写文件

算法实现：用户输入要保存的文件名，打开文件，根据线性表的 $L->\text{size}$ 存入数据空间，将当前的线性表作为文件保存，数据保存完毕，关闭文件。

时空效率分析：时间复杂度为 $O(1)$ ，空间复杂度为 $O(n)$

2. 读取文件

算法实现：用户输入要读取的文件名，打开文件，读取该文件的数据存入当前线性表，直到文件中的数据读入完成，关闭文件。

时空效率分析：时间复杂度为 $O(1)$ ，空间复杂度为 (n)

2.3 系统实现

2.3.1 实验环境

实验环境为 Windows10，编译器版本为 TDM-GCC 4.7.1，代码采用开源的编译器 VS2017 编写。由指定的 MakeFile 来完成编译。

文件说明：

*.vs: VS 系统文件

*LinearList_2.cpp:线性表及演示系统实现

*Debug: 调试文件夹

*LinearList_2: 源码文件夹

*LinearList_2.sln: VS Solution 文件

源码文件里面关键文件说明：

*LinearList_2.h: 链表头文件

*LinearList_2.cpp: 测试代码文件

*data_01,data_02: 存好的线性表文件

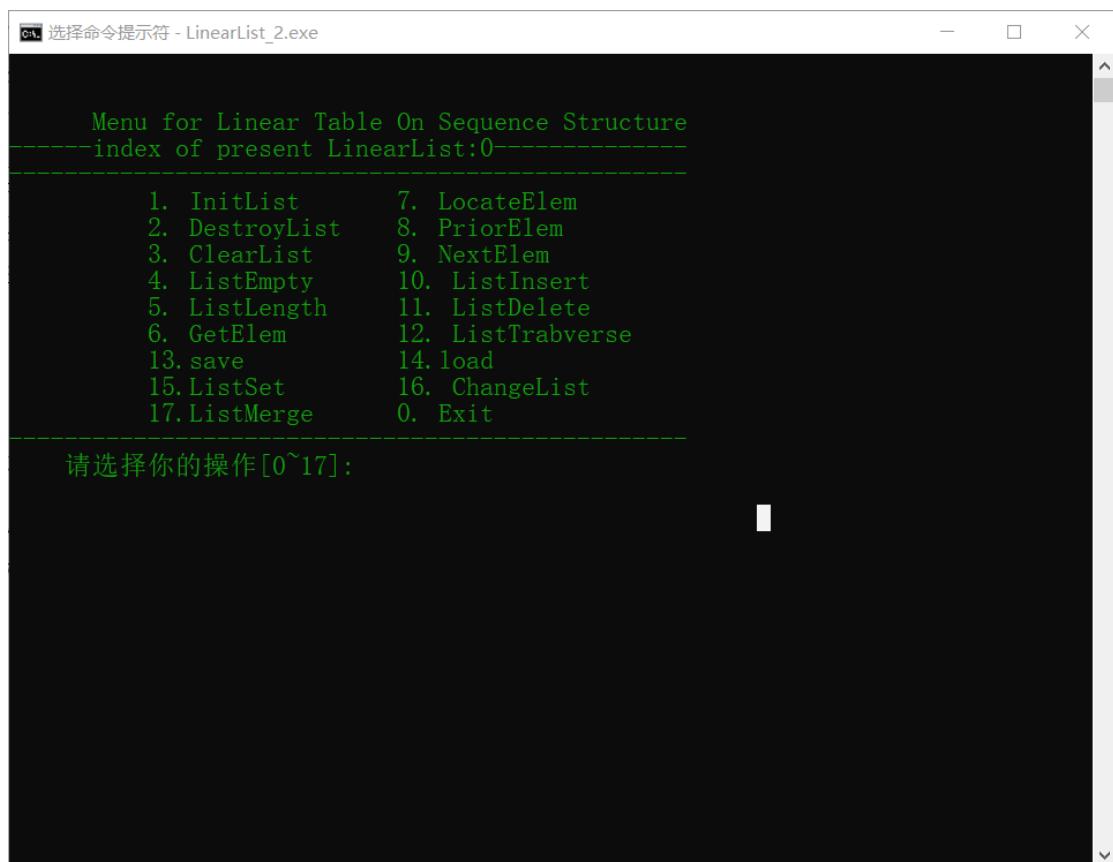
2.3.2 代码亮点

所有代码采用 Google C/C++ 标准代码规范，函数库所有注释采用英文，符合生产规范。错误检查和提示全面，根据不同的错误会有不同的提示信息。由于在 Windows 环境下编程，所有的 API 接口命名采用标准短横线命名模式。

用户每次输入操作都会整齐打印出函数的名称，参数，返回值和作用，而且是用英文，专有名词使用准确不会产生歧义，整个演示系统整洁美观，对用户而言具有操作友好性。

2.3.3 操作演示

界面展示：



1. 首先创建一个线性表

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:1
Your choise:1
/*
*Function Name:InitList
*Module:Data structures
*Parameter:LinkList L
*Return:status
*Use:initialize the LinearList
*/
*LinearList initialize success
```

2. 然后用 ListEmpty 查看，显示这个新建的表是空表

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:4
Your choise:4
/*
*Function Name>ListEmpty
*Module>Data structures
*Parameter:LinkList L
*Return:status
*Use:judge the LinearList null or not
*/
*LinearList is empty
```

3. 然后删除这个表

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:3
Your choise:3
/*
*Function Name:ClearList
*Module:Data structures
*Parameter:LinkList L
*Return:status
*Use:reset the LinearList
*/
*This is a null list
*LinearList reset error
```

4. 再用 ListEmpty 查看，显示线性表不存在

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:4
Your choise:4
/*
*Function Name:ListEmpty
*Module:Data structures
*Parameter:LinkList L
*Return:status
*Use:judge the LinearList null or not
*/
*LinearList is null
```

5. 重新创建一个线性表

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:1
Your choise:1
/*
*Function Name:InitList
*Module:Data structures
*Parameter:LinkList L
*Return:status
*Use:initialize the LinearList
*/
*LinearList initialize success
```

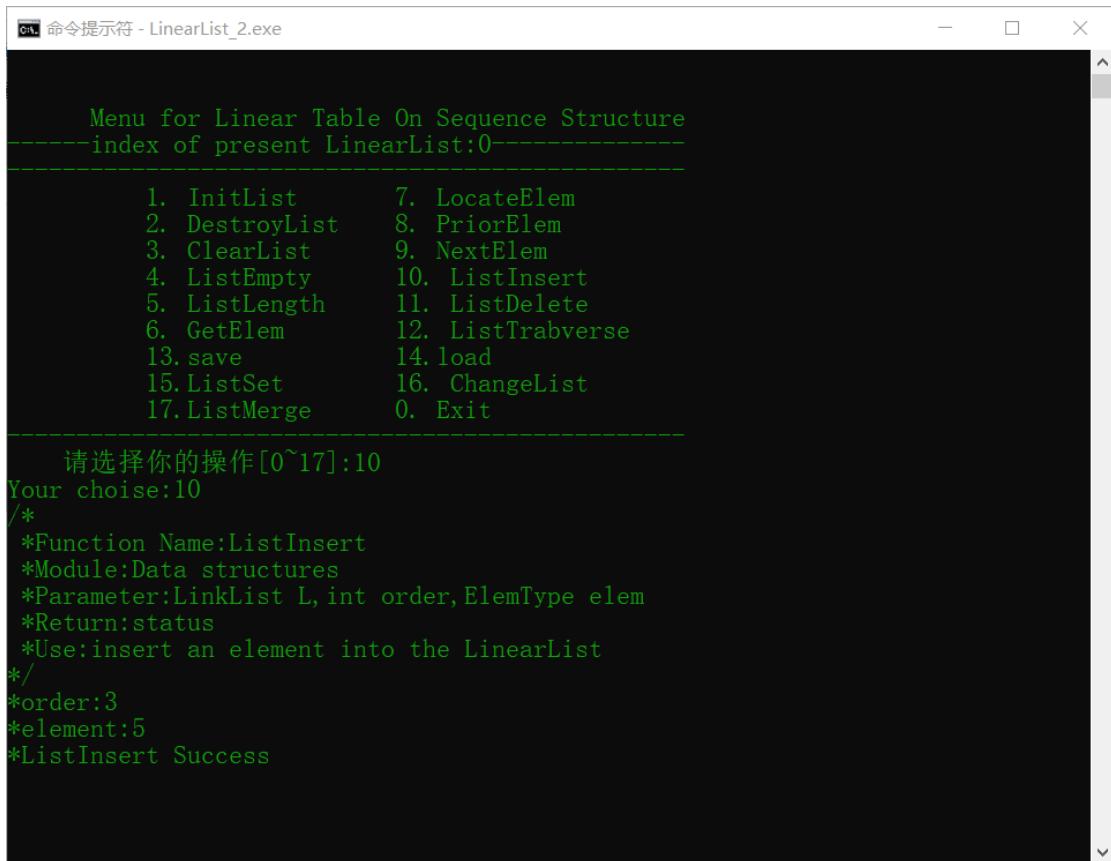
6. 一次插入三个元素 1,3,5

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:10
Your choise:10
/*
*Function Name>ListInsert
*Module>Data structures
*Parameter:LinkList L, int order, ElemtType elem
*Return:status
*Use:insert an element into the LinearList
*/
*order:1
*element:1
*ListInsert Success
```

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

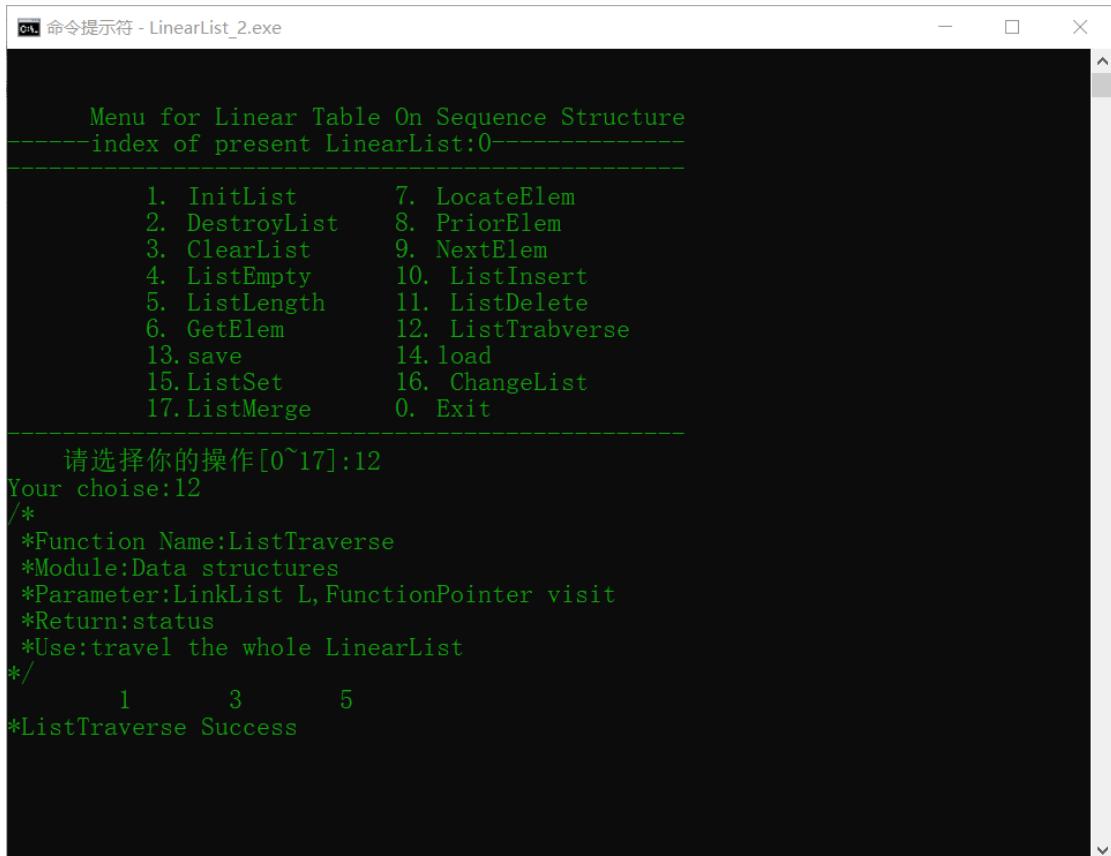
请选择你的操作 [0~17]:10
Your choise:10
/*
*Function Name>ListInsert
*Module>Data structures
*Parameter:LinkList L, int order, ElemtType elem
*Return:status
*Use:insert an element into the LinearList
*/
*order:2
*element:3
*ListInsert Success
```



```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:0
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:10
Your choise:10
/*
*Function Name>ListInsert
*Module>Data structures
*Parameter:LinkList L, int order, ElemtType elem
*Return:status
*Use:insert an element into the LinearList
*/
*order:3
*element:5
>ListInsert Success
```

7. 打印这个表（操作 12 中的 visit 函数是将元素的值打印）



```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:0
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:12
Your choise:12
/*
*Function Name>ListTraverse
*Module>Data structures
*Parameter:LinkList L, FunctionPointer visit
*Return:status
*Use:travel the whole LinearList
*/
      1      3      5
>ListTraverse Success
```

8. 用 ListEmpty 查看，显示这个表是非空表

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge     0. Exit

请选择你的操作 [0~17]:4
Your choise:4
/*
 *Function Name:ListEmpty
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:judge the LinearList null or not
 */
*LinearList is not empty
```

9. 查看这个表的长度，显示是 3

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:5
Your choise:5
/*
*Function Name>ListLength
*Module>Data structures
*Parameter:LinkList L
*Return:int
*Use:return the length of the LinearList
*/
*the length of the LinearList is:3
```

10. 获得这个表中的第 2 个数据元素，显示是 3

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:6
Your choise:6
/*
*Function Name>GetElem
*Module>Data structures
*Parameter:LinkList L, int order, ElemtType& elem
*Return:status
*Use:get the No. order element of the LinearList
*/
*The order:2
*GetElem Success
*The element is:3
```

11. 获得这个表中第一个与 3 相等的数据元素（操作 7 的 compare 函数是判断值是否相等）

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit
-----
请选择你的操作[0~17]:7
Your choise:7
/*
*Function Name:LocateElem
*Module>Data structures
*Parameter:LinkList L, ElemtType elem, FunctionPointer compare
*Return:int
*Use:find the first element in the LinearList
*with the relationship compare with e
*/
*input e:3
*LocateElem Success
*The order of the element is:2
```

12. 获得元素 3 之前的元素

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

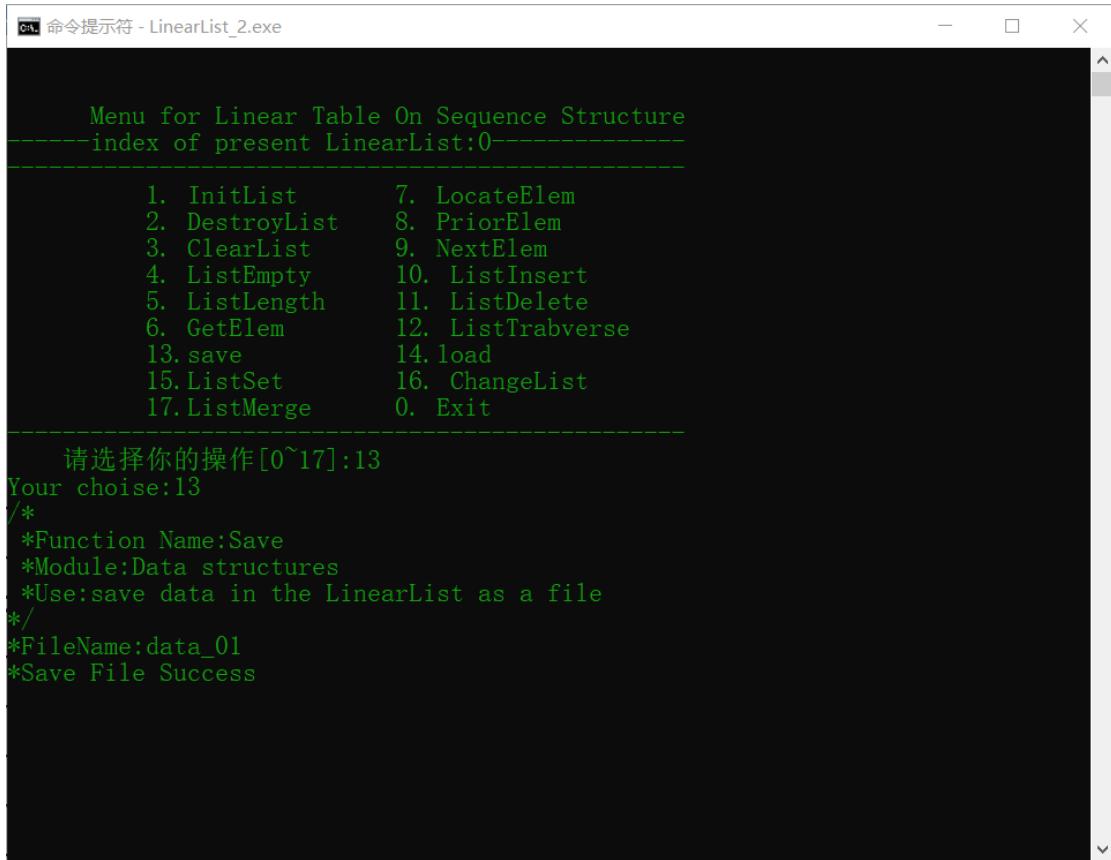
请选择你的操作 [0~17]:8
Your choise:8
/*
*Function Name:PriorElem
*Module:Data structures
*Parameter:LinkList L, ElemtType cur, ElemtType&pre_e
*Return:status
*Use:find the pre element of cur in the LinearList
*/
*cur:3
*PriorElem Success
*the PriorElem of cur is:1
```

13. 获得元素 3 之后的元素

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:9
Your choise:9
/*
*Function Name:NextElem
*Module:Data structures
*Parameter:LinkList L, ElemtType cur, ElemtType&next_e
*Return:status
*Use:find the next element of cur in the LinearList
*/
*cur:3
*NextElem Success
*the NextElem of cur is:5
```

14. 将当前线性表的数据写入“data_01”



```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge     0. Exit

请选择你的操作 [0~17]:13
Your choise:13
/*
 *Function Name:Save
 *Module:Data structures
 *Use:save data in the LinearList as a file
 */
*FileName:data_01
*Save File Success
```

15. 删 除当前表的第 2 个元素

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:11
Your choise:11
/*
*Function Name>ListDelete
*Module>Data structures
*Parameter:LinkList L, int order, ElemtType& elem
*Return:status
*Use:delete the No. order element of the LinearList
*/
*order:2
>ListDelete Success
>The element be deleted is:3
```

16. 打印这个表，发现第二个元素成功删除

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

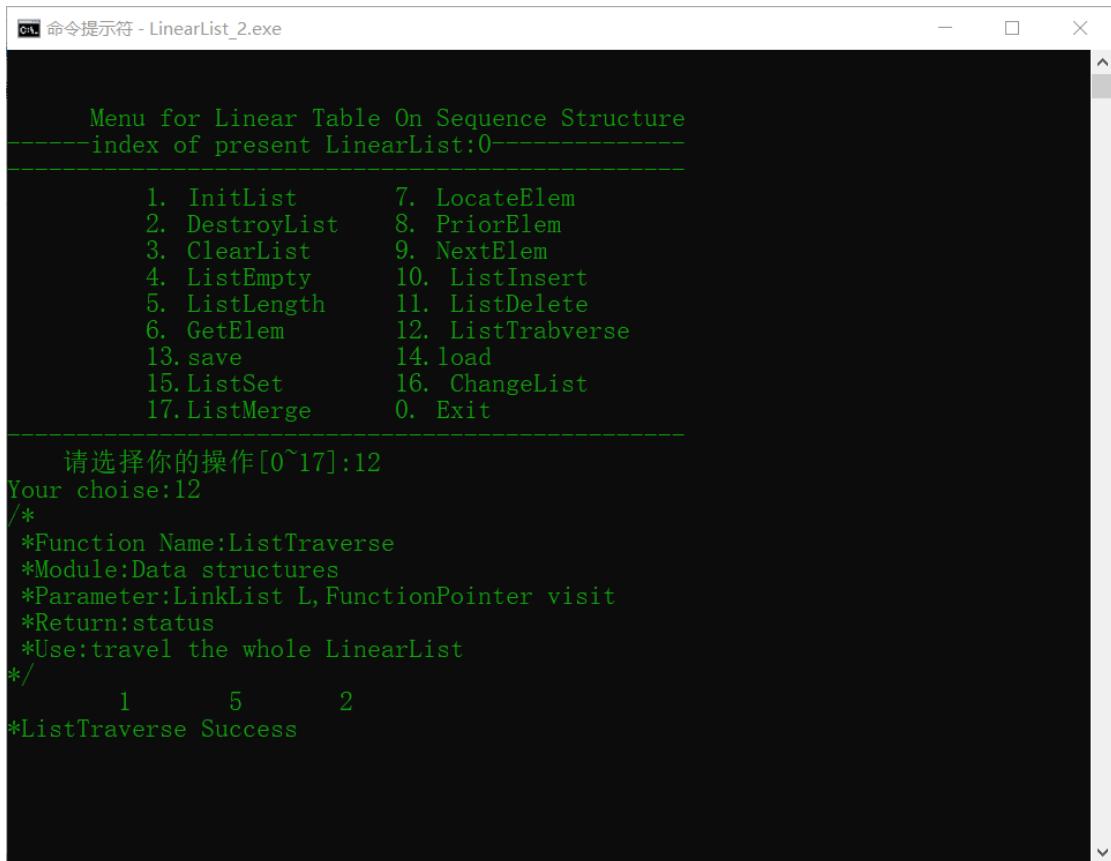
请选择你的操作 [0~17]:12
Your choise:12
/*
*Function Name>ListTraverse
*Module>Data structures
*Parameter:LinkList L, FunctionPointer visit
*Return:status
*Use:travel the whole LinearList
*/
      1      5
>ListTraverse Success
```

17. 在表的第三个位置插入 2

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge     0. Exit

请选择你的操作 [0~17]:10
Your choise:10
/*
*Function Name>ListInsert
*Module>Data structures
*Parameter:LinkList L, int order, ElemtType elem
*Return:status
*Use:insert an element into the LinearList
*/
*order:3
*element:2
*ListInsert Success
```

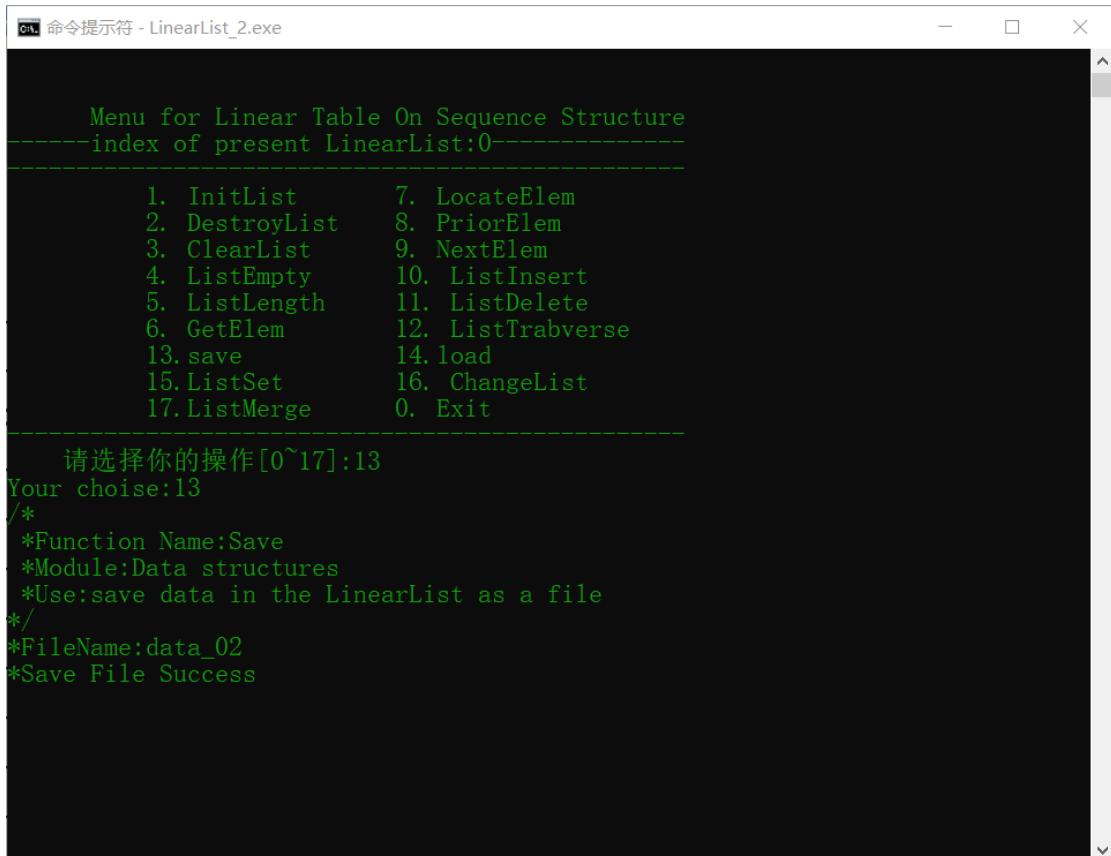
18. 打印这个表，发现插入数据成功



```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:12
Your choise:12
/*
 *Function Name>ListTraverse
 *Module>Data structures
 *Parameter:LinkList L, FunctionPointer visit
 *Return:status
 *Use:travel the whole LinearList
*/
      1      5      2
>ListTraverse Success
```

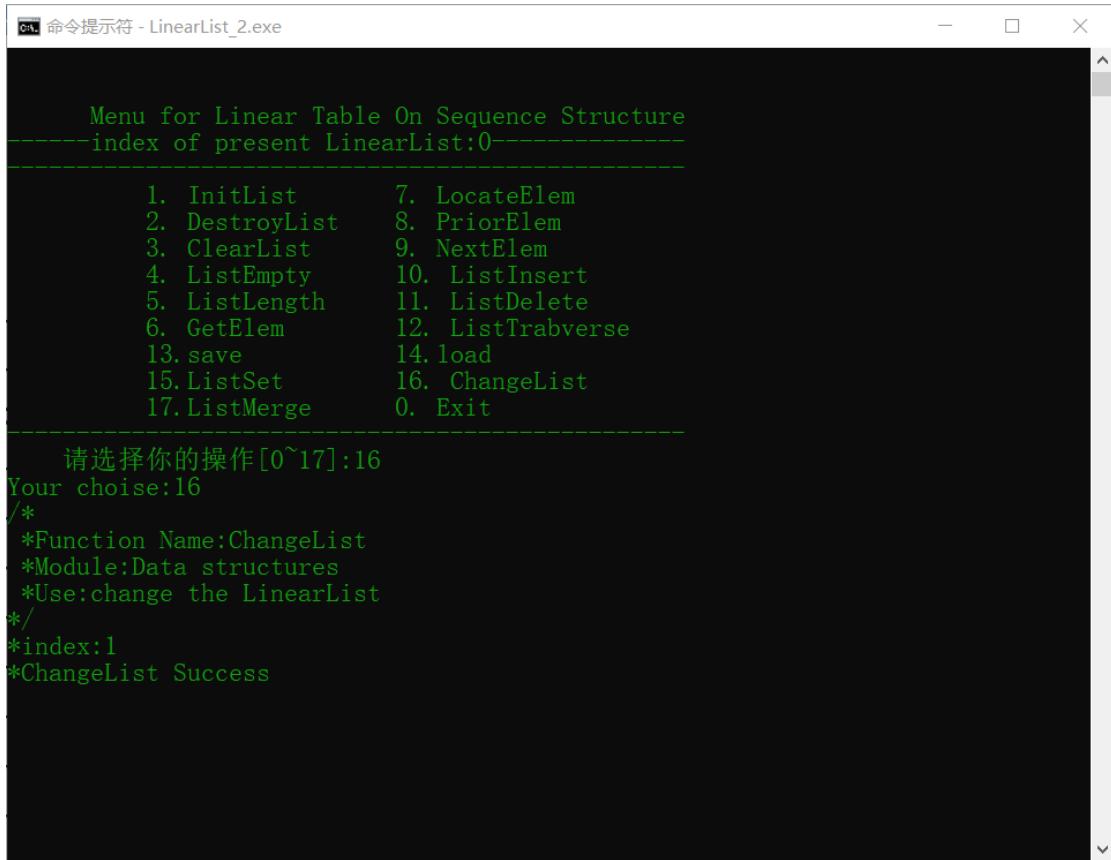
19. 再将当前表的数据写入文件“data_02”



```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:13
Your choise:13
/*
 *Function Name>Save
 *Module>Data structures
 *Use:save data in the LinearList as a file
*/
*FileName:data_02
*Save File Success
```

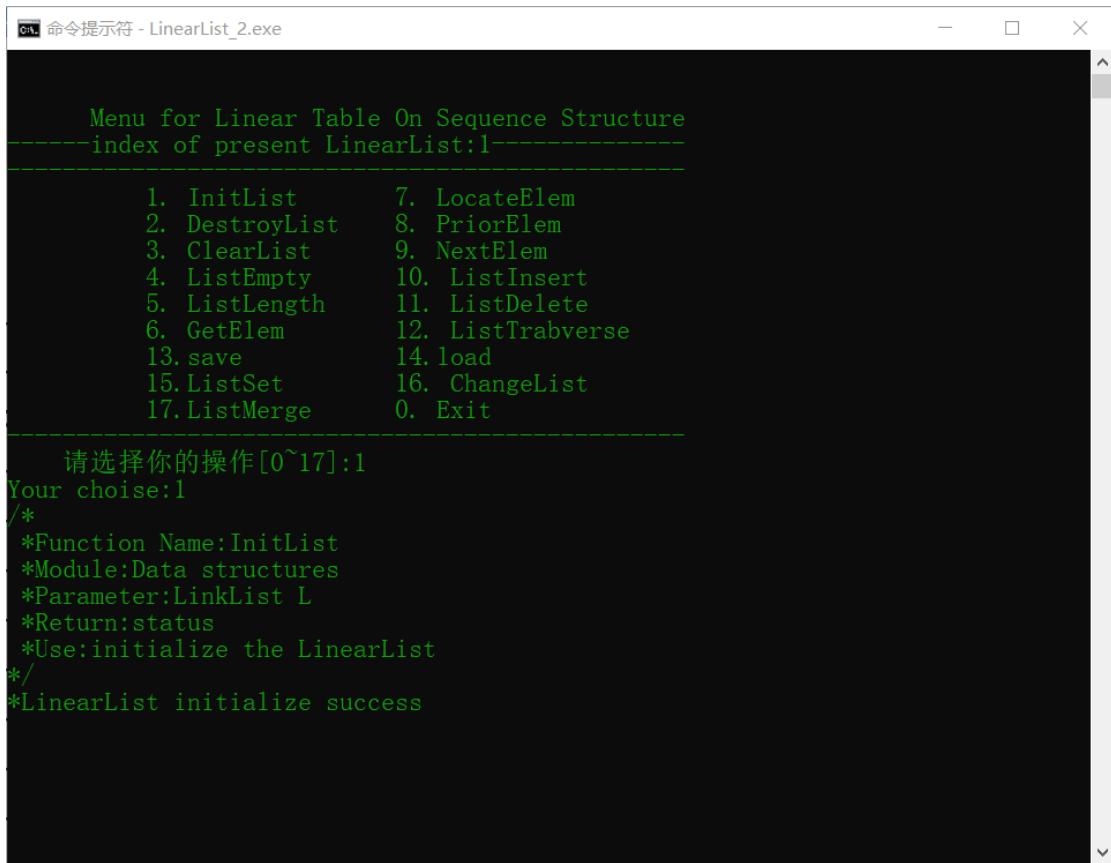
20. 切换线性表，切换索引为 1



```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:16
Your choise:16
/*
 *Function Name:ChangeList
 *Module:Data structures
 *Use:change the LinearList
 */
*index:1
*ChangeList Success
```

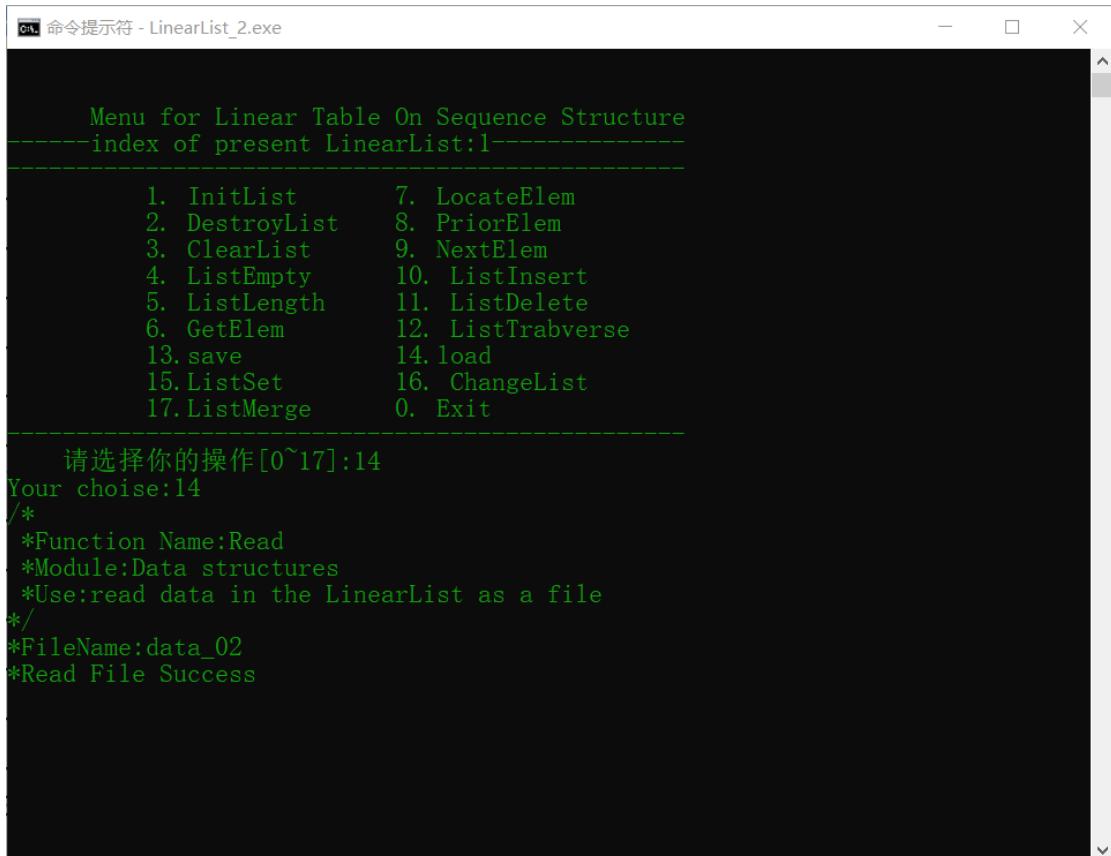
21. 创建一个新表



```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:1
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:1
Your choise:1
/*
 *Function Name:InitList
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:initialize the LinearList
 */
*LinearList initialize success
```

22. 将文件“data_02”的数据加载进当前线性表



```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:1
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:14
Your choise:14
/*
 *Function Name:Read
 *Module:Data structures
 *Use:read data in the LinearList as a file
 */
*FileName:data_02
*Read File Success
```

23. 打印这个表，发现数据被成功加载

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:1
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:12
Your choise:12
/*
*Function Name:ListTraverse
*Module:Data structures
*Parameter:LinkList L, FunctionPointer visit
*Return:status
*Use:travel the whole LinearList
*/
      1      5      2
>ListTraverse Success
```

24. 再将文件“data_01”的数据加载进当前线性表

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:1
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:14
Your choise:14
/*
*Function Name:Read
*Module:Data structures
*Use:read data in the LinearList as a file
*/
*FileName:data_01
*Read File Success
```

25. 打印这个表，发现数据也被成功加载

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:1
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:12
Your choise:12
/*
*Function Name>ListTraverse
*Module>Data structures
*Parameter:LinkList L, FunctionPointer visit
*Return:status
*Use:travel the whole LinearList
*/
      1      3      5
*ListTraverse Success
```

26. 切换回索引为 0 的线性表

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:1
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:16
Your choise:16
/*
*Function Name:ChangeList
*Module:Data structures
*Use:change the LinearList
*/
*index:0
*ChangeList Success
```

27. 打印当前表，显示的数据与 data_02 中的数据相同，说明各线性表之间的操作相互独立

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:12
Your choise:12
/*
*Function Name:ListTraverse
*Module:Data structures
*Parameter:LinkList L, FunctionPointer visit
*Return:status
*Use:travel the whole LinearList
*/
      1      5      2
*ListTraverse Success
```

28. 切换为索引为 1 的线性表

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:16
Your choise:16
/*
*Function Name:ChangeList
*Module:Data structures
*Use:change the LinearList
*/
*index:1
*ChangeList Success
```

29. 将当前线性表进行排序

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:1
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge     0. Exit

请选择你的操作 [0~17]:15
Your choise:15
/*
*Function Name:ListSort
*Module:Data structures
*Parameter:LinkList L
*Return:status
*Use:sort the LinearList
*/
*ListSort Success
```

30. 打印这个表，发现排序成功

```
命令提示符 - LinearList_2.exe
----- Menu for Linear Table On Sequence Structure -----
index of present LinearList:1
----- 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. save          14. load
          15. ListSet        16. ChangeList
          17. ListMerge      0. Exit
----- 请选择你的操作 [0~17]:12
Your choise:12
/*
 *Function Name:ListTraverse
 *Module:Data structures
 *Parameter:LinkList L, FunctionPointer visit
 *Return:status
 *Use:travel the whole LinearList
 */
      1      3      5
*ListTraverse Success
```

31. 切换为索引为 2 的线性表

```
命令提示符 - LinearList_2.exe
----- Menu for Linear Table On Sequence Structure -----
index of present LinearList:1
----- 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. save          14. load
          15. ListSet        16. ChangeList
          17. ListMerge      0. Exit
----- 请选择你的操作 [0~17]:16
Your choise:16
/*
 *Function Name:ChangeList
 *Module:Data structures
 *Use:change the LinearList
 */
*index:2
*ChangeList Success
```

32. 创建一个新表

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:2
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge     0. Exit

请选择你的操作 [0~17]:1
Your choise:1
/*
*Function Name:InitList
*Module:Data structures
*Parameter:LinkList L
*Return:status
*Use:initialize the LinearList
*/
*LinearList initialize success
```

33. 将线性表 0 和线性表 1 合并为线性表 2

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:2
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:17
Your choise:17
/*
*Function Name:ListMerge
*Module:Data structures
*Parameter:LinkList La, LinkList Lb, Linklist Lc
*Return:status
*Use:merge two LinearList as new one
*/
*index of La:0
*index of Lb:1
*index of Lc:2
*ListMerge Success
*ListMerge and ListSort Success
```

34. 打印这个表，发现合并成功并已经完成排序

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:2
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:12
Your choise:12
/*
*Function Name:ListTraverse
*Module:Data structures
*Parameter:LinkList L, FunctionPointer visit
*Return:status
*Use:travel the whole LinearList
*/
          1      1      2      3      5      5
*ListTraverse Success
```

35. 正常退出系统

```
命令提示符
Menu for Linear Table On Sequence Structure
index of present LinearList:2
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge     0. Exit

请选择你的操作 [0~17]:0
Welcom next time
c:\Users\SKTT1Ryze\Desktop\DataStructure_Code\LinearList_2\Debug>
```

通过以上的演示证明，我们的线性表的功能是完整实现了的，可以演示要求

的十二种线性表上的运算，并且做到了多线性表管理，实现了文件存储功能，还

有线性表排序和线性表合并功能，不存在内存泄漏，完整的达到了实验要求。

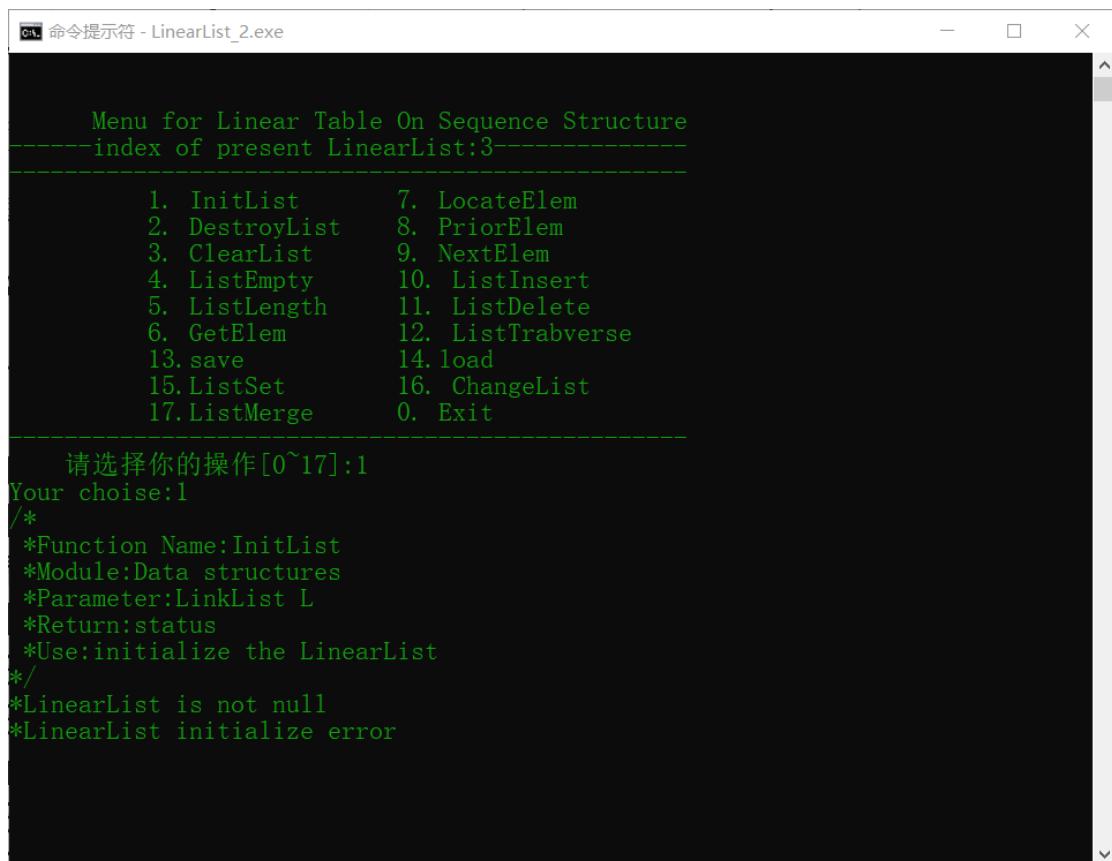
2.3.4 系统测试

在上面操作的基础上，我们再创建两个线性表，索引分别为 3 和 4，索引为 3 的线性表中的数据元素是 1,4,2,7,5；索引为 4 的线性表中的数据元素是

12,4,10,3,6；分别写入文件“data_03”和“data_04”

下面以这两个线性表基础，进行系统测试。

1. 初始化已存在的线性表



```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:3
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:1
Your choise:1
/*
 *Function Name:InitList
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:initialize the LinearList
 */
*LinearList is not null
*LinearList initialize error
```

2. 删除不存在的线性表

```
命令提示符 - LinearList_2.exe

Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:2
Your choise:2
/*
 *Function Name:DestroyList
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:destroy the LinearList
 */
*The LinearList do not exist
*LinearList destroy error
```

3. 清空不存在的线性表

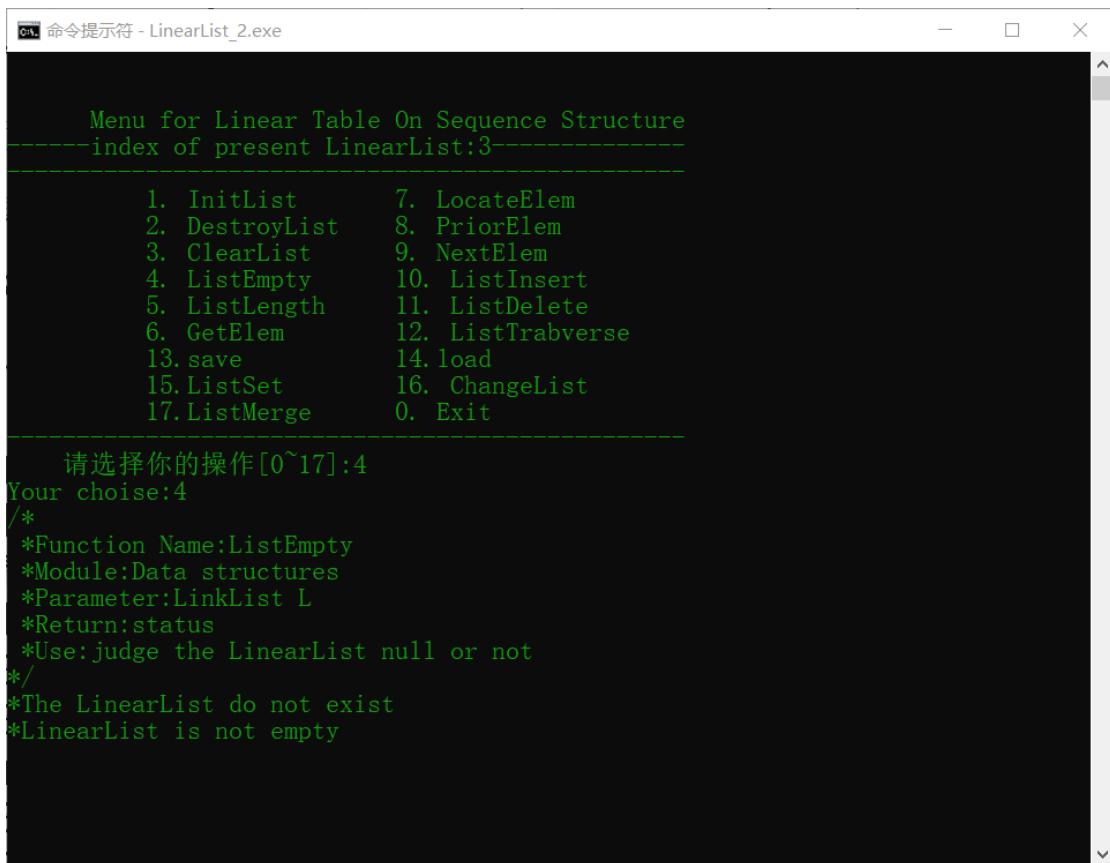
```
命令提示符 - LinearList_2.exe

Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:3
Your choise:3
/*
 *Function Name:ClearList
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:reset the LinearList
 */
*The LinearList do not exist
*LinearList reset error
```

4. 判断不存在的线性表是否为空



```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:3
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:4
Your choise:4
/*
 *Function Name:ListEmpty
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:judge the LinearList null or not
 */
 *The LinearList do not exist
 *LinearList is not empty
```

5. 获得不存在的线性表的表长

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:5
Your choise:5
/*
 *Function Name>ListLength
 *Module>Data structures
 *Parameter:LinkList L
 *Return:int
 *Use:return the length of the LinearList
 */
*The LinearList do not exist
*the length of the LinearList is:0
```

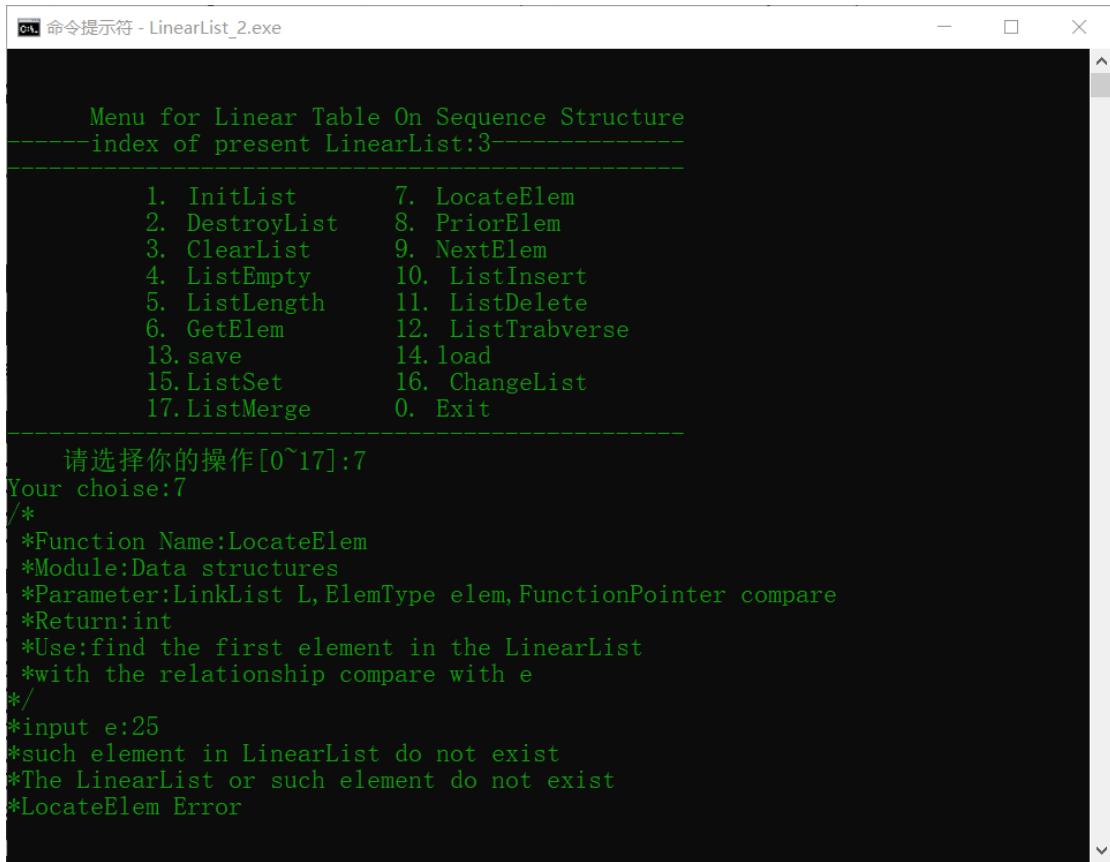
6. 获得表中不存在的元素

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:6
Your choise:6
/*
 *Function Name>GetElem
 *Module>Data structures
 *Parameter:LinkList L, int order, ElemtType& elem
 *Return:status
 *Use:get the No. order element of the LinearList
 */
*The order:11
*No. i element do not exist
*GetElem Error
```

7. 确定表中不存在的元素的位置



```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:3
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge     0. Exit

请选择你的操作[0~17]:7
Your choise:7
/*
 *Function Name:LocateElem
 *Module>Data structures
 *Parameter:LinkList L, ElemtType elem, FunctionPointer compare
 *Return:int
 *Use:find the first element in the LinearList
 *with the relationship compare with e
 */
*input e:25
*sueh element in LinearList do not exist
*The LinearList or such element do not exist
*LocateElem Error
```

8. 获得表中不存在的元素的前元素

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:8
Your choise:8
/*
 *Function Name:PriorElem
 *Module:Data structures
 *Parameter:LinkList L, ElemtType cur, ElemtType&pre_e
 *Return:status
 *Use:find the pre element of cur in the LinearList
 */
*cur:9
*cur is not in LinearList
*PriorElem Error
```

9. 获得表中第一个元素的前元素

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:8
Your choise:8
/*
 *Function Name:PriorElem
 *Module:Data structures
 *Parameter:LinkList L, ElemtType cur, ElemtType&pre_e
 *Return:status
 *Use:find the pre element of cur in the LinearList
 */
*cur:1
*cur is the first element
*PriorElem Error
```

10. 获得表中不存在的元素的后元素

```
命令提示符 - LinearList_2.exe
-----
Menu for Linear Table On Sequence Structure
index of present LinearList:3
-----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:9
Your choise:9
/*
 *Function Name:NextElem
 *Module:Data structures
 *Parameter:LinkList L, ElemtType cur, ElemtType&next_e
 *Return:status
 *Use:find the next element of cur in the LinearList
 */
*cur:11
*cur is not in LinearList
*NextElem Error
```

11. 获得表中最后一个元素的后元素

```
命令提示符 - LinearList_2.exe

Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:9
Your choise:9
/*
 *Function Name:NextElem
 *Module:Data structures
 *Parameter:LinkList L, ElemtType cur, ElemtType&next_e
 *Return:status
 *Use:find the next element of cur in the LinearList
 */
*cur:5
*cur is the last element
*NextElem Error
```

12. 插入元素不合法

```
命令提示符 - LinearList_2.exe

Menu for Linear Table On Sequence Structure
index of present LinearList:3

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:10
Your choise:10
/*
 *Function Name>ListInsert
 *Module>Data structures
 *Parameter:LinkList L, int order, ElemtType elem
 *Return:status
 *Use:insert an element into the LinearList
 */
*order:12
*element:1
*the value of i is overflow
>ListInsert Error
```

13. 删 除 表 中 不 存 在 的 元 素

```
命令提示符 - LinearList_2.exe
Menu for Linear Table On Sequence Structure
index of present LinearList:3
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:11
Your choise:11
/*
*Function Name>ListDelete
*Module>Data structures
*Parameter:LinkList L, int order, Elemtyp& elem
*Return:status
*Use:delete the No. order element of the LinearList
*/
*order:12
*the value of i is overflow
*ListDelete Error
```

14. 打 印 不 存 在 的 线 性 表

```
命令提示符 - LinearList_2.exe

Menu for Linear Table On Sequence Structure
index of present LinearList:5

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:12
Your choise:12
/*
*Function Name:ListTraverse
*Module:Data structures
*Parameter:LinkList L, FunctionPointer visit
*Return:status
*Use:travel the whole LinearList
*/
*The LinearList do not exist
*ListTraverse Error
```

15. 为不存在的线性表排序

```
命令提示符 - LinearList_2.exe

Menu for Linear Table On Sequence Structure
index of present LinearList:5

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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作 [0~17]:15
Your choise:15
/*
*Function Name:ListSort
*Module:Data structures
*Parameter:LinkList L
*Return:status
*Use:sort the LinearList
*/
*The LinearList do not exist
*ListSort Error
```

16. 线性表合并中有线性表不存在

```

命令提示符 - LinearList_2.exe

Menu for Linear Table On Sequence Structure
----- index of present LinearList:5 -----
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. save         14. load
15. ListSet      16. ChangeList
17. ListMerge    0. Exit

请选择你的操作[0~17]:17
Your choise:17
/*
 *Function Name>ListMerge
 *Module>Data structures
 *Parameter:LinkList La, LinkList Lb, Linklist Lc
 *Return:status
 *Use:merge two LinearList as new one
 */
*index of La:3
*index of Lb:4
*index of Lc:5
*The LinearList do not exist

```

2.4 实验小结

这次的实验内容是基于上次实验的改进，改为链式存储结构，虽然难度上比第一次实验有所提升，但是由于链式存储结构在日常开发中很常用，队列，树我大多都用链式存储实现，所以我对这次的实验内容还是很熟悉的。

本系统完整地实现了课程要求的全部功能，实现了多线性表管理和文件存储功能，系统健壮性良好，可以输出各种情况下的错误提示，并且在整个实验过程中避免了内存泄漏，甚至还加上附加功能排序和合并线性表，较好的达到了预期效果。

3 基于二叉链表的二叉树实现

3.1 问题描述

3.1.1 二叉树抽象数据类型

依据最小完备性和常用性相结合的原则，设计了二叉树的数据对象和数据
类

系，并定义了二叉树的初始化二叉树、销毁二叉树、置空二叉树、判定二
叉树是否为空树、求二叉树深度和获得指定元素等 20 种基本运算，具体数据
和运算功能定义如下。

(1) 创建二叉树：函数名称是 CreateBiTree(T,definition)；初始条件是
definition 给出二叉树 T 的定义，如带空子树的二叉树前序遍历序列、或前序+
中序、或后序+中序；操作结果是按 definition 构造二叉树 T。

注： 1 * GB3 要求 T 中各结点关键字具有唯一性。后面各操作的实现，也都要满足
一棵二叉树中关键字的唯一性，不再赘述； 2 * GB3CreateBiTree 中根据 definition 生成
T，不应在 CreateBiTree 中输入二叉树的定义。

(2) 销毁二叉树：函数名称是 DestroyBiTree(T)；初始条件是二叉树 T 已存在；
操作结果是销毁二叉树 T。

(3) 清空二叉树：函数名称是 ClearBiTree (T)；初始条件是二叉树 T 存在；
操作结果是将二叉树 T 清空。

(4) 判定空二叉树：函数名称是 BiTreeEmpty(T)；初始条件是二叉树 T 存在；

操作结果是若 T 为空二叉树则返回 TRUE，否则返回 FALSE。

(5) 求二叉树深度：函数名称是 BiTreeDepth(T)；初始条件是二叉树 T 存在；

操作结果是返回 T 的深度。

(6) 查找结点：函数名称是 LocateNode(T,e)；初始条件是二叉树 T 已存在，

e 是和 T 中结点关键字类型相同的给定值；操作结果是返回查找到的结点指针，

如无关键字为 e 的结点，返回 NULL。

(7) 结点赋值：函数名称是 Assign(T,e,value)；初始条件是二叉树 T 已存在，

e 是和 T 中结点关键字类型相同的给定值；操作结果是关键字为 e 的结点赋值为

value。

(8) 获得兄弟结点：函数名称是 GetSibling(T,e)；初始条件是二叉树 T 存在，

e 是和 T 中结点关键字类型相同的给定值；操作结果是返回关键字为 e 的结点的

(左或右) 兄弟结点指针。若关键字为 e 的结点无兄弟，则返回 NULL。

(9) 插入结点：函数名称是 InsertNode(T,e,LR,c)；初始条件是二叉树 T 存在，

e 是和 T 中结点关键字类型相同的给定值，LR 为 0 或 1，c 是待插入结点；操作

结果是根据 LR 为 0 或者 1，插入结点 c 到 T 中，作为关键字为 e 的结点的左或

右孩子结点，结点 e 的原有左子树或右子树则为结点 c 的右子树。

(10) 删除结点：函数名称是 DeleteNode(T,e)；初始条件是二叉树 T 存在，e 是和 T 中结点关键字类型相同的给定值。操作结果是删除 T 中关键字为 e 的结点；同时，如果关键字为 e 的结点度为 0，删除即可；如关键字为 e 的结点度为 1，用关键字为 e 的结点孩子代替被删除的 e 位置；如关键字为 e 的结点度为 2，用 e 的左孩子代替被删除的 e 位置，e 的右子树作为 e 的左子树中最右结点的右子树。

(11) 前序遍历：函数名称是 PreOrderTraverse(T,Visit())；初始条件是二叉树 T 存在，Visit 是一个函数指针的形参（可使用该函数对结点操作）；操作结果：先序遍历，对每个结点调用函数 Visit 一次且一次，一旦调用失败，则操作失败。

注：前序、中序和后序三种遍历算法，要求至少一个用非递归算法实现。

(12) 中序遍历：函数名称是 InOrderTraverse(T,Visit))；初始条件是二叉树 T 存在，Visit 是一个函数指针的形参（可使用该函数对结点操作）；操作结果是中序遍历 t，对每个结点调用函数 Visit 一次且一次，一旦调用失败，则操作失

败。

13 /* GB2 后序遍历：函数名称是 PostOrderTraverse(T,Visit))；初始条件是二叉树 T 存在， Visit 是一个函数指针的形参（可使用该函数对结点操作）；操作结果是后序遍历 t，对每个结点调用函数 Visit 一次且一次，一旦调用失败，则操作失败。

(14) 按层遍历：函数名称是 LevelOrderTraverse(T,Visit))；初始条件是二叉树 T 存在， Visit 是对结点操作的应用函数；操作结果是层序遍历 t，对每个结点调用函数 Visit 一次且一次，一旦调用失败，则操作失败。

3.1.2 多二叉树的管理

我设计了一个包装结构体指针数组，将已存在的所有二叉树用结构体指针数组存储，并对每个二叉树赋予独特的 ID 来标识，在每次操作中通过 ID 来选择要进行操作的结构体指针

3.1.3 演示系统和文件存储的设计

参考附录的框架，并添加了一些元素，该框架将完成函数调用所需实参值的准备和函数执行结果的实现，并给出适当的操作提示显示，整体以命令行呈

现。

此外设计的文件存储，选择了二进制文件存储的方式。一个树存为一个文件，用户可以自行选择存储。存储的实现是先将树的前序和中序遍历按顺序存在一个文件中，读取文件的时候根据文件中的前序和中序顺序创建二叉树。

3.2 系统设计

3.2.1 数据物理结构

1.二叉树的存储数据结构

```
//the main struct of binary tree
typedef struct BinaryTree
{
    int id;//ID of BITree
    int size;//numbers of Nodes
    struct BiTNode *root;//root of BiTree
}BinaryTree, *BinaryTreePos;
```

2.二叉树节点的存储数据结构

```
typedef struct BiTNode
{
    KeyType index;
    ElemType value;
    struct BiTNode *lchild, *rchild;//left and right child
```

```
}BiTNode, *BiTPos;//define of Node ADT
```

3.2.2 演示系统

演示系统包括用户操作界面和功能调用部分。

演示系统界面语言为英文，所有操作和提示语言均为英文。

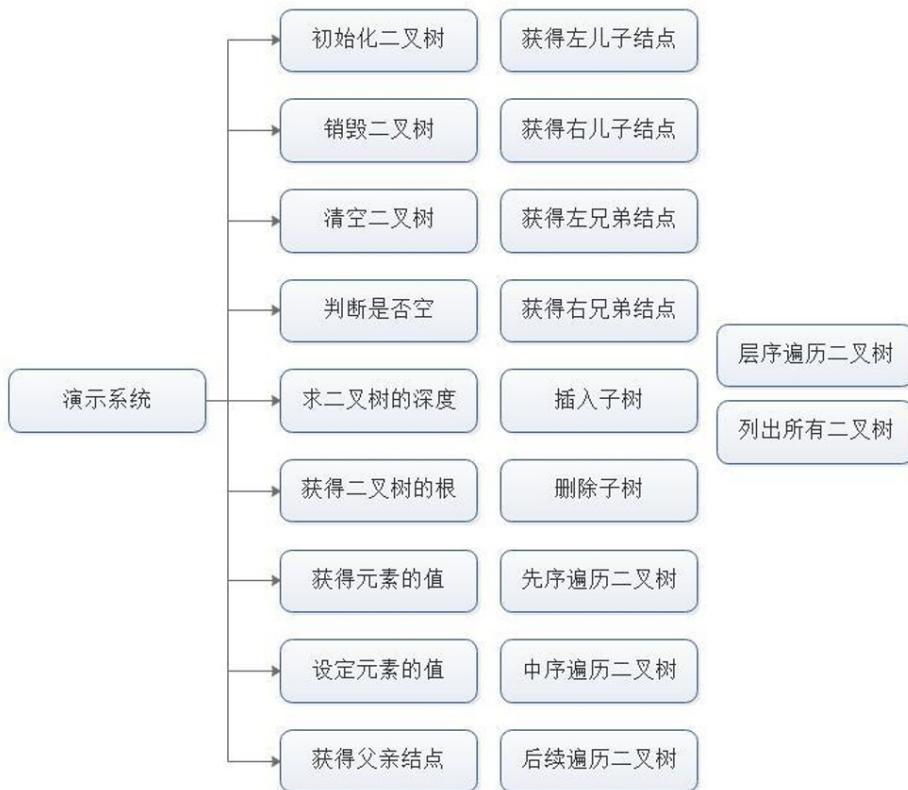
用户操作界面输出可选的线性表操作，用户输入数字选择要进行的操作。

在

用户选择操作后，功能调用部分会显示函数的名称，参数，返回值和作用，
系统

提示用户输入参数。

功能调用部分将用户输入的有关信息传递给线性数据结构的操作函数进行
调用，并对函数的返回值进行处理判断输出相应的提示信息。



3.2.3 二叉树运算演示算法

1. status CreateBiTree(BinaryTreePos&T, int definition);

功能：选择四种方法之一创建二叉树

算法实现：

(1) 前序加中序：

寻找前序遍历的第一个节点在中序遍历中的位置，其左侧为该节点的左子树，右侧为右子树，然后在左子树部分递归此操作，再在右子树部分递归此操作

(2) 后序加中序：

寻找后序遍历的最后一个节点在中序遍历中的位置，其左侧为该节点的左子树，右侧为右子树，然后在左子树部分递归此操作，再在右子树部分递归此操作

(3) 前序加空节点：

输入一个节点，若为空节点，则将对应节点赋值为 NULL，否则为对应节点申请内存空间，赋值 value，然后在该节点的左子树递归此操作，再在该节点的右子树递归此操作

(4) 后序加空节点：

将输入存在一个数组中，其中空节点用 0 标识，然后从数组后面向前遍历，如果遇到 0，则将对应节点赋值为 NULL，否则为对应节点申请内存空间，赋值 value，然后在该节点的右子树递归此操作，再在该节点的左子树递归此操作

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

2.status DestroyBiTree(BinaryTreePos&T);

功能：删除二叉树

算法实现：free 掉二叉树所有节点，再 free 掉二叉树结构体本身，放回 OK

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

3.status ClearBiTree(BinaryTreePos&T);

功能：清空二叉树

算法实现：递归 free 掉二叉树所有节点，返回 OK

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

4.status BiTreeEmpty(BinaryTreePos&T);

功能：判断二叉树是否为空

算法实现：如果 $T->root==NULL$ ，返回 TRUE，否则返回 FALSE

时空效率分析：时间复杂度为 $O(1)$ ，空间复杂度为 $O(1)$

5.int BiTreeDepth(BiTPos&root);

功能：求二叉树的深度

算法实现：对任意一个节点，返回其左子树的深度和右子树的深度中大的那个加一。递归执行这个操作，最后返回深度。

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

6.BiTPos LocateNode(BinaryTreePos&T, KeyType e);

功能：在二叉树中查找一个节点

算法实现：从根节点开始，层遍历查找，若找到，返回其值，否则打印找不到的信息，返回 NULL

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

7.status Assign(BinaryTreePos&T, KeyType e, ElemtType value);

功能：设置指定节点的值

算法实现：同 6 一样，先找到这个节点，再给其 value 赋值，若找到返回 OK，否则打印找不到的信息，返回 ERROR

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

8.BiTPos GetSibling(BinaryTreePos&T, KeyType e);

功能：获得指定节点的兄弟节点

算法实现：先层遍历寻找到指定节点的父节点，并返回该节点是其父节点的左节点还是右节点，然后返回父节点的另外一个节点的指针

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

9.status InsertNode(BinaryTreePos&T, KeyType e, int LR, BiTPos&c);

功能：插入节点

算法实现：先层遍历找到指定节点，根据 LR 等于 0 或 1 将节点 c 设置为指定节点的左儿子或者右儿子，指定节点原来的左儿子或者右儿子被设置为节点 c 的右儿子，若未找到返回 ERROR，否则返回 OK

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

10.status DeleteNode(BinaryTreePos&T, KeyType e);

功能：删除节点

算法实现：层遍历找到指定节点和指定节点的父节点，若指定节点的度为 0，则直接删除掉指定节点，若指定节点的度为 1，则将指定节点的子节点作为其父节点的子节点，替换它原来的位置，删除指定节点，若指定节点的度为 2，则将指定节点的左节点作为其父节点的节点，替换它原来的位置，指定节点的右节点作为其左子树最右边的节点的右节点。若找到指定节点，则返回 OK，否则返回 ERROR

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

11.statusPreOrderTraverse(BinaryTreePos&T,status(*visit)
(BiTPos&node));

功能：前序遍历二叉树中的每个节点

算法实现：采用栈来进行前序遍历

- (1) 根节点进栈
- (2) 在栈为空之前进行循环
- (3) 循环操作为：遍历栈顶元素，若出栈的节点有左节点，左节点进

栈，若有右节点，右节点进栈

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

12.statusInOrderTraverse(BinaryTreePos&T,status(*visit)
(BiTPos&node));

功能：中序遍历二叉树中的每个节点

算法实现：采用递归来进行遍历

- (1) 在该节点左子树上进行中序遍历
- (2) 遍历该节点
- (3) 在该节点右子树上进行中序遍历

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

13.statusPostOrderTraverse(BinaryTreePos&T,status(*visit)
(BiTPos&node));

功能：后序遍历二叉树中的每个节点

算法实现：采用递归来进行遍历

- (1) 在该节点左子树上进行后序遍历
- (2) 在该节点右子树上进行后序遍历
- (3) 遍历该节点

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

14.statusLevelOrderTraverse(BinaryTreePos&T,status(*visit))

(BiTPos&node));

功能：层遍历二叉树中的每个节点

算法实现：采用队列来进行层序遍历

(1) 根节点进队列

(2) 若队列不为空，进行循环

(3) 循环操作为：队首出列，其左儿子和右儿子分别进队列，遍历出列的节点

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

15.status Show(BinaryTreePos&T);

功能：树形打印二叉树

算法实现：层遍历二叉树，用一个数组存储要打印的顺序，空节点用 N 进行标识，求出二叉树的深度后，在每层打印对应节点的关键字

时空效率分析：时间复杂度为 $O(n)$ ，空间复杂度为 $O(n)$

3.3 系统实现

3.3.1 实验环境

实验环境为 Windows10，编译器版本为 TDM-GCC 4.7.1，代码采用开源的编译器 Dev-C++ 编写。由指定的 MakeFile 来完成编译。

文件说明：

*BiTreeADT.h：二叉树库头文件

*main.cpp:二叉树及演示系统实现
*BinaryTree_dev.dev:DEV 文件
*BinaryTree_dev.exe:可执行文件
*BinaryTree_dev.layout:LAYOUT 文件
*Makefile.win:自动化编译命令
*Temp_Tree:二叉树的临时存储文件
Tree:二叉树的存储文件

3.3.2 代码亮点

所有代码采用 Google C/C++ 标准代码规范，函数库所有注释采用英文，符合生产规范。错误检查和提示全面，根据不同的错误会有不同的提示信息。由于在 Windows 环境下编程，所有的 API 接口命名采用标准短横线命名模式。

用户每次输入操作都会整齐打印出函数的名称，参数，返回值和作用，而且是用英文，专有名词使用准确不会产生歧义，整个演示系统整洁美观，对用户而言具有操作友好性。

3.3.3 操作演示

界面演示

Windows PowerShell

```
Menu for Linear Table On Sequence Structure
index of present LinearList:0
-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:
```

1. 方式一创建树

Windows PowerShell

```
请选择你的操作 [-4~17]:1
Your choise:1
/*
 *Function Name:CreateBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T, int definition
 *Return:status
 *Use:Create a tree
*/
*Definition:
*1. preorder and inorder
*2. postorder and inorder
*3. preorder and nullnode
*4. postorder and nullnode
1
*Input positive pre_order, end with -1:
index:0
1
*Input success
index:1
2
*Input success
index:2
4
*Input success
index:3
5
*Input success
index:4
3
```

```
Windows PowerShell
*Input success
index:6
-1
*Input end
*Input positive in_order, end with -1:
index:0
4
*Input success
index:1
2
*Input success
index:2
5
*Input success
index:3
1
*Input success
index:4
3
*Input success
index:5
6
*Input success
index:6
-1
*Input end
*Create Success
*size:6
*Operator Success
```

2. 树形打印

```
请选择你的操作 [-4~17] : -2
*Pre0derTravel Success
*In0derTravel Success
=====1=====
=====
=====2=====3=====
=====
=====4=====5=====6=====
=====
*Destroy Success
*Pre0derTravel Success
```

3. 删除树

```
: 请选择你的操作 [-4~17] : 2
Your choise:2
/*
 *Function Name:DestroyBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:status
 *Use:destroy the BinaryTree
 */
*Destroy Success
*Operator Success
```

4. 方式二创建树

```
Windows PowerShell
请选择你的操作[-4~17]:1
Your choise:1
/*
 *Function Name:CreateBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T, int definition
 *Return:status
 *Use:Create a tree
*/
*Definition:
*1.preorder and inorder
*2.postorder and inorder
*3.preorder and nullnode
*4.postorder and nullnode
2
*Input positive post_order, end with -1:
index:0
4
*Input success
index:1
5
*Input success
index:2
2
*Input success
index:3
6
*Input success
index:4
3

Windows PowerShell
*Input success
index:6
-1
*Input end
*Input positive in_order, end with -1:
index:0
4
*Input success
index:1
2
*Input success
index:2
5
*Input success
index:3
1
*Input success
index:4
3
*Input success
index:5
6
*Input success
index:6
-1
*Input end
*Create Success
*size:6
*Operator Success
```

5. 树形打印

```
请选择你的操作 [-4~17] : -2
*PreOderTravel Success
*InOderTravel Success
=====1=====
=====
=====2=====3=====
=====
=====4=====5=====6=====
=====
*Destroy Success
*PreOderTravel Success
```

6. 删除树

```
: 请选择你的操作 [-4~17] : 2
Your choise:2
/*
 *Function Name:DestroyBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:status
 *Use:destroy the BinaryTree
 */
*Destroy Success
*Operator Success
```

7. 方式三创建树

```
*input success
2
*input success
4
*input success
0
*input success
0
*input success
5
*input success
0
*input success
0
*input success
3
*input success
0
*input success
6
*input success
0
*input success
0
*LevelOrderTraverse Success
*Create Success
*size:6
*Operator Success
```

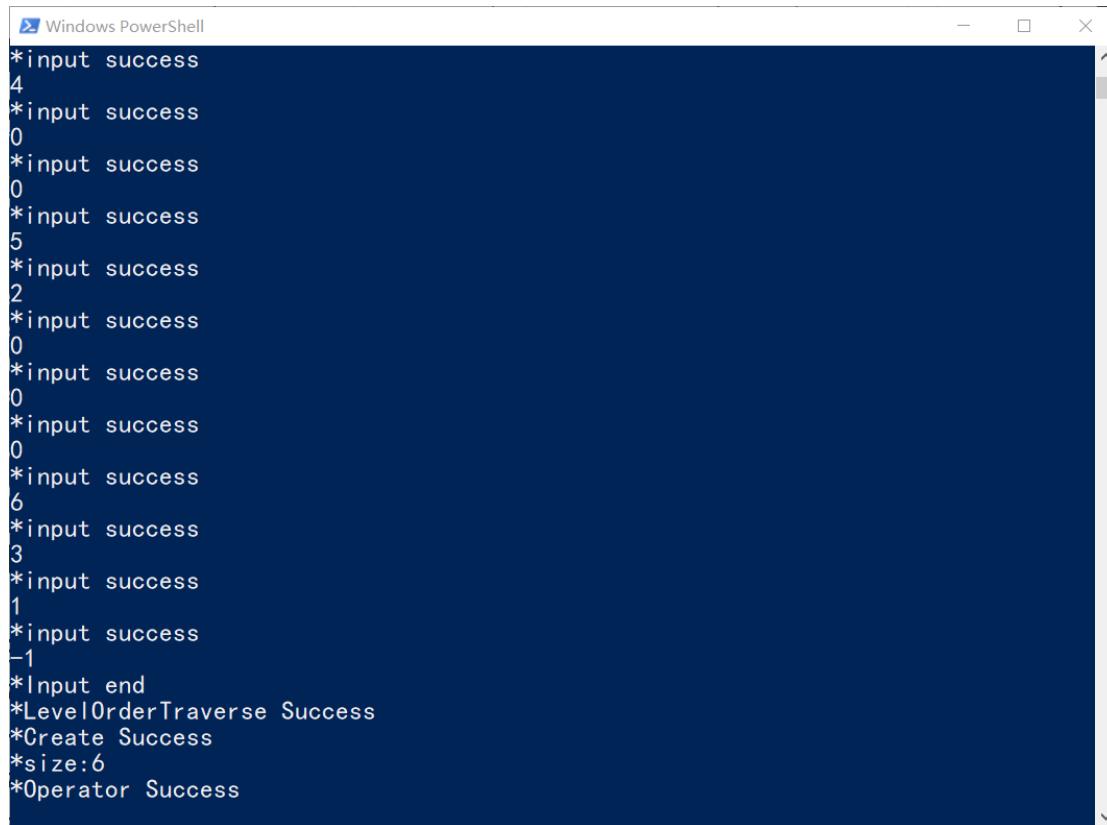
8. 树形打印

```
请选择你的操作[-4~17]:-2
*Pre0derTravel Success
*In0derTravel Success
=====
1
=====
2 3
=====
4 5 6
=====
*Destroy Success
*Pre0derTravel Success
```

9. 删除树

```
: 请选择你的操作[-4~17]:2
Your choise:2
/*
 *Function Name:DestroyBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:status
 *Use:destroy the BinaryTree
 */
*Destroy Success
*Operator Success
```

10. 方式四创建树



```
*input success
4
*input success
0
*input success
0
*input success
5
*input success
2
*input success
0
*input success
0
*input success
6
*input success
3
*input success
1
*input success
-1
*Input end
*LevelOrderTraverse Success
*Create Success
*size:6
*Operator Success
```

11. 树形打印



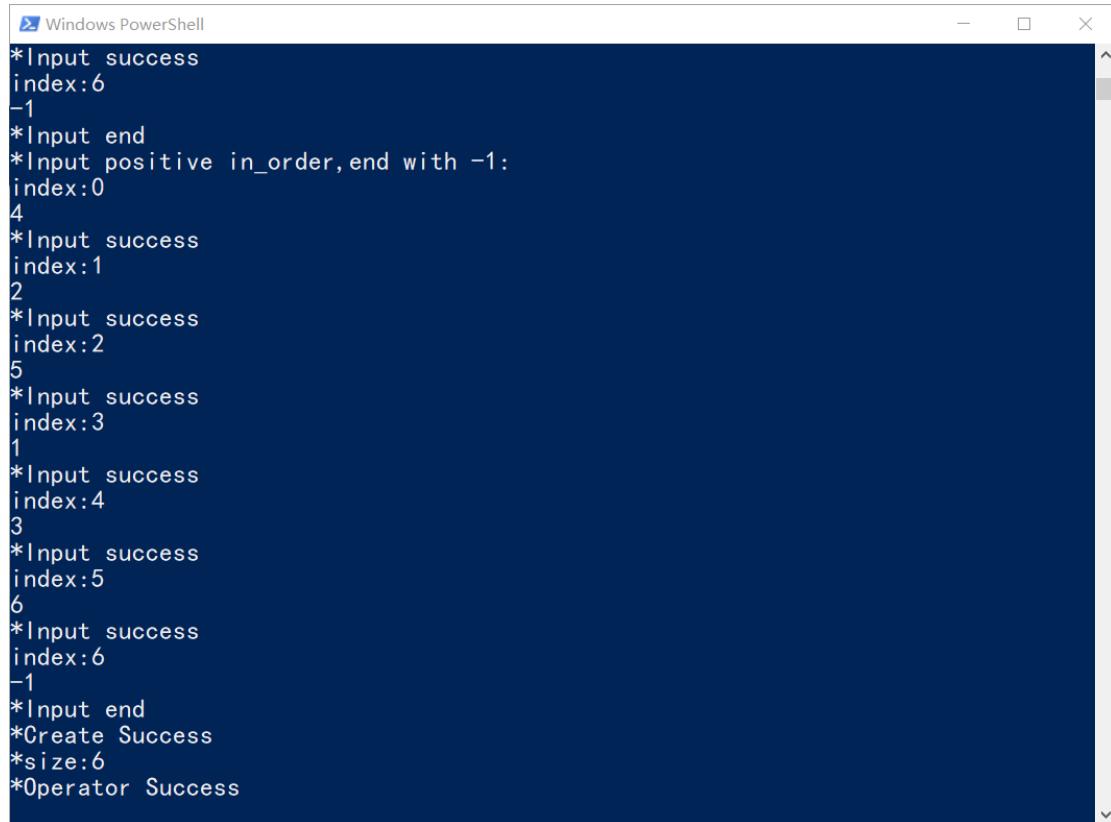
```
请选择你的操作 [-4~17] : -2
*Pre0derTravel Success
*In0derTravel Success
-----1-----
-----2-----3---
-----4---5---6-
-----
*Destroy Success
*Pre0derTravel Success
```

12. 删除树

```
: 请选择你的操作 [-4~17]:2
Your choise:2
/*
 *Function Name:DestroyBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:status
 *Use:destroy the BinaryTree
 */
*Destroy Success
*Operator Success
```

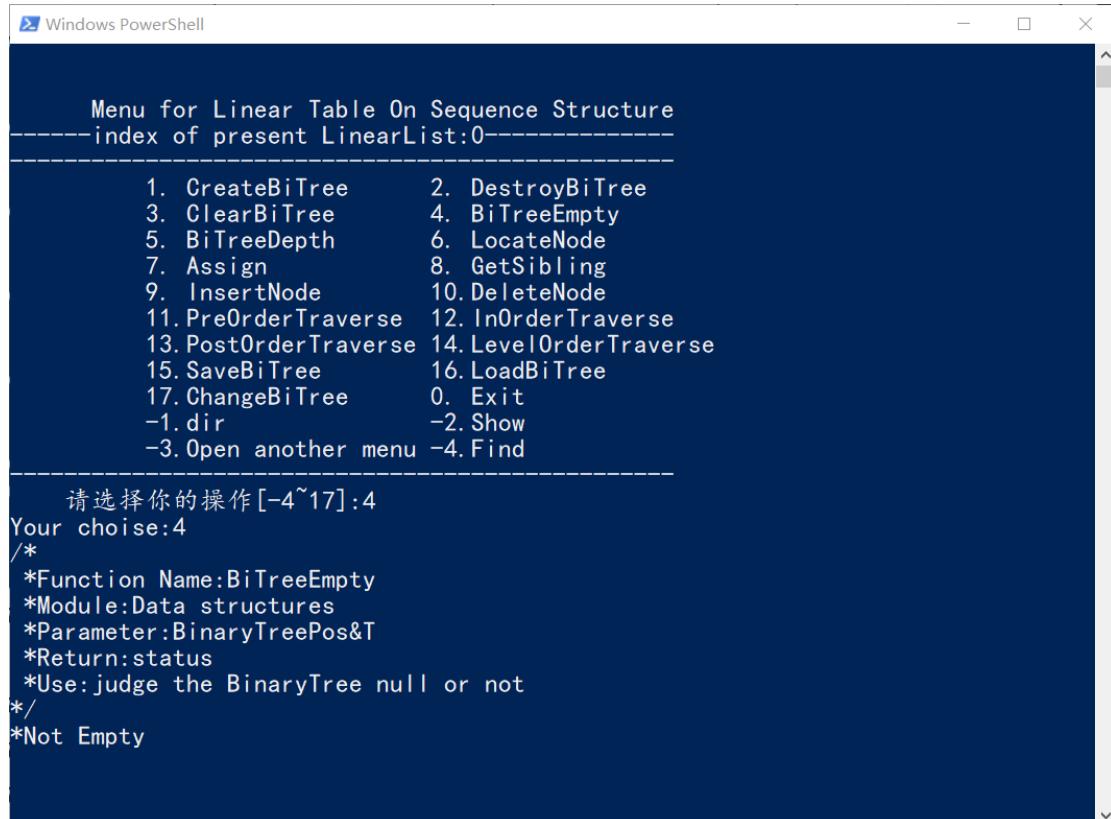
13. 方式一创建树

```
Windows PowerShell
: 请选择你的操作 [-4~17]:1
Your choise:1
/*
 *Function Name>CreateBiTree
 *Module>Data structures
 *Parameter:BinaryTreePos&T, int definition
 *Return:status
 *Use>Create a tree
*/
*Definition:
*1. preorder and inorder
*2. postorder and inorder
*3. preorder and nullnode
*4. postorder and nullnode
1
*Input positive pre_order, end with -1:
index:0
1
*Input success
index:1
2
*Input success
index:2
4
*Input success
index:3
5
*Input success
index:4
3
```



```
*Input success
index:6
-1
*Input end
*Input positive in_order,end with -1:
index:0
4
*Input success
index:1
2
*Input success
index:2
5
*Input success
index:3
1
*Input success
index:4
3
*Input success
index:5
6
*Input success
index:6
-1
*Input end
*Create Success
*size:6
*Operator Success
```

14. 查看树是否为空



```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:4
Your choise:4
/*
*Function Name:BiTreeEmpty
*Module:Data structures
*Parameter:BinaryTreePos&T
*Return:status
*Use:judge the BinaryTree null or not
*/
*Not Empty
```

15. 查看树的深度

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:5
Your choise:5
/*
 *Function Name:BiTreeDepth
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:int
 *Use:return the depth of the BinaryTree
 */
*Depth:3
```

16. 清空树

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:3
Your choise:3
/*
 *Function Name:ClearBiTree
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:reset the LinearList
 */
*Operator Success
```

17. 查看树是否为空

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:4
Your choise:4
/*
 *Function Name:BiTreeEmpty
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:status
 *Use:judge the BinaryTree null or not
 */
*Empty
```

18. 方式一创建一个一样的树

```
Windows PowerShell

请选择你的操作 [-4~17]:1
Your choise:1
/*
 *Function Name:CreateBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T, int definition
 *Return:status
 *Use>Create a tree
 */
*Definition:
*1. preorder and inorder
*2. postorder and inorder
*3. preorder and nullnode
*4. postorder and nullnode
2
*Input positive post_order, end with -1:
index:0
4
*Input success
index:1
5
*Input success
index:2
2
*Input success
index:3
6
*Input success
index:4
3
```

```
*Input success
index:6
-1
*Input end
*Input positive in_order,end with -1:
index:0
4
*Input success
index:1
2
*Input success
index:2
5
*Input success
index:3
1
*Input success
index:4
3
*Input success
index:5
6
*Input success
index:6
-1
*Input end
*Create Success
*size:6
*Operator Success
```

19. 查找关键字为 2 的节点

```
Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:6
Your choise:6
/*
 *Function Name:LocateNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:BiTPos
 *Use:get the node with key e in the BiTree
 */
*Input the Key:2
*Found
*Operator Success
*key:2 value:0
```

20. 为关键字为 2 的节点赋值为 2

```
Windows PowerShell
-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir                -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:7
Your choise:7
/*
*Function Name:Assign
*Module:Data structures
*Parameter:BinaryTreePos&T, KeyType e, ElemtType value
*Return:int
*Use:find the node with key e in the BiTree
*and change the value
*/
*Input the Key:2
*Input the value:
2
*Found
*value change
*Operator Success
```

21. 获得关键字为 2 的节点的兄弟节点和关键字为 6 的节点的兄弟节点

```
Windows PowerShell
-----
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir                -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:8
Your choise:8
/*
*Function Name:GetSibling
*Module:Data structures
*Parameter:BinaryTreePos&T, KeyType e
*Return:BiTPos
*Use:find the node with key e in BiTree
*/
*if node have lchild,return,else if have rchild
*/
*return,else return null
*/
*Input the Key:2
*Found
*Find Father
*have right brother
*Operator Success
*key:3 value:0
```

```
Windows PowerShell
3. ClearBiTree      4. BiTreeEmpty
5. BiTreeDepth       6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:8
Your choise:8
/*
 *Function Name:GetSibling
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:BiTPos
 *Use:find the node with key e in BiTree
 */
 *if node have lchild,return,else if have rchild
 */
 *return,else return null
 */
 *Input the Key:6
 *Found
 *Find Father
 *have no brother
 *Operator Error
```

22. 先序遍历

```
Windows PowerShell
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree        4. BiTreeEmpty
5. BiTreeDepth         6. LocateNode
7. Assign             8. GetSibling
9. InsertNode          10. DeleteNode
11. PreOrderTraverse   12. InOrderTraverse
13. PostOrderTraverse  14. LevelOrderTraverse
15. SaveBiTree          16. LoadBiTree
17. ChangeBiTree        0. Exit
-1. dir                -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:11
Your choise:11
/*
 *Function Name:PreOrderTraverse
 *Module:Data structures
 *Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
 *Return:status
 *Use:Travel the BinaryTree in pre_order
 */
key:1 value:0
key:2 value:2
key:4 value:0
key:5 value:0
key:3 value:0
key:6 value:0
*PreOrderTravel Success
*Operator Success
```

23. 中序遍历

```
Windows PowerShell
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3. Open another menu -4. Find
-----
请选择你的操作 [-4~17]:12
Your choise:12
/*
 *Function Name:InOrderTraverse
 *Module:Data structures
 *Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
 *Return:status
 *Use:Travel the BinaryTree in in_order
*/
key:4 value:0
key:2 value:2
key:5 value:0
key:1 value:0
key:3 value:0
key:6 value:0
*In0derTravel Success
*Operator Success
```

24. 后序遍历

```
Windows PowerShell
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3. Open another menu -4. Find
-----
请选择你的操作 [-4~17]:13
Your choise:13
/*
 *Function Name:PostOrderTraverse
 *Module:Data structures
 *Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
 *Return:status
 *Use:Travel the BinaryTree in post_order
key:4 value:0
key:5 value:0
key:2 value:2
key:6 value:0
key:3 value:0
key:1 value:0
*Post0derTravel Success
*Operator Success
```

25. 层遍历

```
Windows PowerShell
-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:14
Your choise:14
/*
*Function Name:LevelOrderTraverse
*Module:Data structures
*Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
*Return:status
*Use:Travel the BinaryTree in level_order
key:1 value:0
key:2 value:2
key:3 value:0
key:4 value:0
key:5 value:0
key:6 value:0
*LevelOrderTraverse Success
*Operator Success
```

26. 在关键字为 3 的节点插入左节点

```
Windows PowerShell
-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:9
Your choise:9
/*
*Function Name:InsertNode
*Module:Data structures
*Parameter:BinaryTreePos&T, KeyType e, int LR, BiTPos&c
*Return:status
*Use:insert a node
*/
*input the value of the node:7
*Input the Key:3
*LR:0 or 1: 0
*Found
*have right child
*Insert Success
*Operator Success
```

27. 树形打印

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17] : -2
*PreOrderTravel Success
*InOrderTravel Success
=====1=====
=====
=====2=====3=====
=====4=====5=====7=====
=====
=====6=====
*Destroy Success
*PreOrderTravel Success
```

28. 在关键字为 2 的节点插入右节点

```
Windows PowerShell

1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17] : 9
Your choise:9
/*
 *Function Name:InsertNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e, int LR, BiTPos&c
 *Return:status
 *Use:insert a node
 */
*input the value of the node:8
*Input the Key:2
*LR:0 or 1: 1
*Found
*have left and right children
*Insert Success
*Operator Success
```

29. 树形打印

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17] : -2
*PreOrderTravel Success
*InOrderTravel Success
=====1=====
=====
=====2=====3=====
=====
=====8=====7=====
=====
=====4=====5=====6=====
=====
*Destroy Success
*PreOrderTravel Success
```

30. 在关键字为 6 的节点插入左节点

```
Windows PowerShell

1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17] : 9
Your choise:9
/*
 *Function Name:InsertNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e, int LR, BiTPos&c
 *Return:status
 *Use:insert a node
 */
*input the value of the node:9
*Input the Key:6
*LR:0 or 1: 0
*Found
*have no child
*Insert Success
*Operator Success
```

31. 树形打印

```
Windows PowerShell
----- index of present LinearList:0 -----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir               -2. Show
-3. Open another menu -4. Find
-----请选择你的操作[-4~17]:-2
*PreOrderTravel Success
*InOrderTravel Success
=====1=====
=====2=====3=====
=====8=====7=====
=====4=====5=====6=====
=====9=====
*Destroy Success
*PreOrderTravel Success
```

32. 删除关键字为 4 的节点

```
Windows PowerShell
----- 11. PreOrderTraverse 12. InOrderTraverse -----
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree          16. LoadBiTree
17. ChangeBiTree         0. Exit
-1. dir                  -2. Show
-3. Open another menu -4. Find
-----请选择你的操作[-4~17]:10
Your choise:10
/*
 *Function Name:DeleteNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:status
 *Use:delete the node with key e
*/
*Input the key:4
*Found
*Find Father
*don't have children
*Delete Success
*key --
*key --
*key --
*key --
*key --
*PostOrderTravel Success
*Key change success
*Operator Success
```

33. 树形打印

```
Windows PowerShell
----- index of present LinearList:0 -----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir                -2. Show
-3. Open another menu -4. Find
-----
请选择你的操作[-4~17]:-2
*PreOrderTravel Success
*InOrderTravel Success
=====
=====1=====
=====
=====2=====3=====
=====
7-----6-----
=====
4-----5-----
=====
8=====
=====
*Destroy Success
*PreOrderTravel Success
```

34. 删除关键字为 6 的节点

```
Windows PowerShell
----- index of present LinearList:0 -----
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir                -2. Show
-3. Open another menu -4. Find
-----
请选择你的操作[-4~17]:10
Your choise:10
/*
 *Function Name:DeleteNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:status
 *Use:delete the node with key e
*/
*Input the key:6
*Found
*Find Father
*have right child
*Delete Success
*key --
*key --
*PostOrderTravel Success
*Key change success
*Operator Success
```

35. 树形打印

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:-2
*PreOrderTravel Success
*InOrderTravel Success
=====1=====
=====
=====2=====3=====
=====
=====6=====5=====
=====
=====4=====7=====
=====
*Destroy Success
*PreOrderTravel Success
```

36. 删除根结点

```
Windows PowerShell

17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:10
Your choise:10
/*
 *Function Name:DeleteNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:status
 *Use:delete the node with key e
 */
*Input the key:1
*Found
*have left and right children
*move right
*move right
*move end
*Delete Success
*key --
*key --
*key --
*key --
*key --
*key --
*PostOrderTravel Success
*Key change success
*Operator Success
```

37. 树形打印

```
Windows PowerShell
Menu for Linear Table On Sequence Structure
index of present LinearList:0
1. CreateBiTree    2. DestroyBiTree
3. ClearBiTree    4. BiTreeEmpty
5. BiTreeDepth    6. LocateNode
7. Assign          8. GetSibling
9. InsertNode     10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree     16. LoadBiTree
17. ChangeBiTree   0. Exit
-1. dir           -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:-2
*PreOrderTravel Success
*InOrderTravel Success
=====
=====
=====
=====
*Destroy Success
*PreOrderTravel Success
```

38. 保存二叉树

```
Windows PowerShell
-----index of present LinearList:0-----
1. CreateBiTree    2. DestroyBiTree
3. ClearBiTree    4. BiTreeEmpty
5. BiTreeDepth    6. LocateNode
7. Assign          8. GetSibling
9. InsertNode     10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree     16. LoadBiTree
17. ChangeBiTree   0. Exit
-1. dir           -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:15
Your choise:15
/*
 *Function Name:SaveBiTree
 *Module:Data structures
 *Use:save the BiTree as file
*/
*FileName:Tree_report
*PreOrderTravel Success
*save_pre success
1, 2      5, 8      3, 0      2, 0      4, 0      6, 9      -1, 9
*InOrderTravel Success
*save_in success
1, 2      5, 8      3, 0      6, 9      4, 0      2, 0      -1, 0
*Save File Success
```

39. 查看内存中存在的二叉树文件，发现成功保存

```
Windows PowerShell
15. SaveBiTree      16. LoadBiTree
17. ChangeBiTree    0. Exit
-1. dir             -2. Show
-3. Open another menu -4. Find
-----
请选择你的操作 [-4~17]:-1
驱动器 C 中的卷是 OS
卷的序列号是 FE68-2CB7

C:\users\SKTT1Ryze\Desktop\DataStructure_Code\BinaryTree_dev 的目录

2019/12/10 21:38 <DIR> .
2019/12/10 21:38 <DIR> ..
2019/11/25 09:40 1,132 BinaryTree_dev.dev
2019/12/02 10:03 760,615 BinaryTree_dev.exe
2019/12/10 13:30 269 BinaryTree_dev.layout
2019/12/02 10:03 30,613 BiTreeADT.h
2019/11/29 22:18 18,968 main.cpp
2019/12/02 10:03 48,120 main.o
2019/12/02 10:03 906 Makefile.win
2019/12/10 21:37 112 Temp_Tree
2019/11/28 21:25 112 Tree_01
2019/11/28 21:51 128 Tree_02
2019/11/28 22:53 112 Tree_03
2019/11/30 00:03 144 Tree_04
2019/11/30 00:08 176 Tree_05
2019/12/10 21:38 112 Tree_report ←
14 个文件     861,519 字节
2 个目录 42,733,506,560 可用字节
```

40. 切换二叉树

```
Windows PowerShell
-----
Menu for Linear Table On Sequence Structure
-----index of present LinearList:0-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir              -2. Show
-3. Open another menu -4. Find
-----
请选择你的操作 [-4~17]:17
Your choise:17
/*
 *Function Name:ChangeBiTree
 *Module:Data structures
 *Use:chage BiTree
 */
*Index:1
*Change Success
```

41. 读取刚刚保存的二叉树

```

Windows PowerShell
----- index of present LinearList:1 -----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir                -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:16
Your choise:16
/*
 *Function Name:LoadBiTree
 *Module:Data structures
 *Use:load the BiTree from a file
 */
*FileName:Tree_report
*pre_load success
*in_load success
1,2      5,8      3,0      2,0      4,0      6,9
*Load Success
*PreOrderTravel Success
*Assign Value Success
*Create Success

```

42. 树形打印

```

Windows PowerShell
----- Menu for Linear Table On Sequence Structure -----
----- index of present LinearList:0 -----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir                -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:-2
*PreOrderTravel Success
*InOrderTravel Success
-----|-----
-----|-----5-----
-----|-----6-----
-----|-----2-----
-----|-----4-----
-----|-----6-----
*Destroy Success
*PreOrderTravel Success

```

3.4 系统测试

1. 删除不存在的树

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:2
Your choise:2
/*
 *Function Name:DestroyBiTree
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:status
 *Use:destroy the BinaryTree
 */
*The BiTree is NULL
*Operator Error
```

2.清空不存在的树

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:3
Your choise:3
/*
 *Function Name:ClearBiTree
 *Module:Data structures
 *Parameter:LinkList L
 *Return:status
 *Use:reset the LinearList
 */
*The BiTree is null
*Operator Error
```

3.判断不存在的树是否为空

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree    2. DestroyBiTree
3. ClearBiTree    4. BiTreeEmpty
5. BiTreeDepth     6. LocateNode
7. Assign          8. GetSibling
9. InsertNode      10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree     16. LoadBiTree
17. ChangeBiTree   0. Exit
-1.dir            -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:4
Your choise:4
/*
 *Function Name:BiTreeEmpty
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:status
 *Use:judge the BinaryTree null or not
 */
*The BiTree is NULL
*Not Empty
```

4.求不存在的树的深度

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree    2. DestroyBiTree
3. ClearBiTree    4. BiTreeEmpty
5. BiTreeDepth     6. LocateNode
7. Assign          8. GetSibling
9. InsertNode      10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree     16. LoadBiTree
17. ChangeBiTree   0. Exit
-1.dir            -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:5
Your choise:5
/*
 *Function Name:BiTreeDepth
 *Module:Data structures
 *Parameter:BinaryTreePos&T
 *Return:int
 *Use:return the depth of the BinaryTree
 */
*The BinaryTree is NULL
```

6.获得不存在的树的节点

```
Windows PowerShell

----- Menu for Linear Table On Sequence Structure -----
index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:6
Your choise:6
/*
 *Function Name:LocateNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:BiTPos
 *Use:get the node with key e in the BiTree
 */
*Input the Key:1
*The BiTree is NULL
*Operator Error
```

7.设置不存在的树的节点的值

```
Windows PowerShell

----- index of present LinearList:1 -----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:7
Your choise:7
/*
 *Function Name:Assign
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e, ElemtType value
 *Return:int
 *Use:find the node with key e in the BiTree
 *and change the value
 */
*Input the Key:1
*Input the value:
1
*The BiTree is NULL
*Operator Error
```

8.获得不存在的树的节点的兄弟节点

```
Windows PowerShell

1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:8
Your choise:8
/*
 *Function Name:GetSibling
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:BiTPos
 *Use:find the node with key e in BiTree
 */
 *if node have lchild,return,else if have rchild
 */
 *return,else return null
 */
 *Input the Key:1
 *The BiTree is NULL
 *Operator Error
```

9. 在不存在的树中插入节点

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----

1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:9
Your choise:9
/*
 *Function Name:InsertNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e, int LR, BiTPos&c
 *Return:status
 *Use:insert a node
 */
 *input the value of the node:1
 *Input the Key:1
 *LR:0 or 1: 1
 *The BiTree is NULL
 *Operator Error
```

10. 删除不存在的树的节点

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:10
Your choise:10
/*
 *Function Name:DeleteNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:status
 *Use:delete the node with key e
 */
*Input the key:1
*The BiTree is NULL
*Operator Error
```

11. 不存在的树的先序遍历，中序遍历，后序遍历和层遍历

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:11
Your choise:11
/*
 *Function Name:PreOrderTraverse
 *Module:Data structures
 *Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
 *Return:status
 *Use:Travel the BinaryTree in pre_order
 */
*The BiTree is null
*Operator Error
```

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:12
Your choise:12
/*
 *Function Name:InOrderTraverse
 *Module:Data structures
 *Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
 *Return:status
 *Use:Travel the BinaryTree in in_order
 */
*The BiTree is null
*Operator Error
```

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:13
Your choise:13
/*
 *Function Name:PostOrderTraverse
 *Module:Data structures
 *Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
 *Return:status
 *Use:Travel the BinaryTree in post_order
 */
*The BiTree is null
*Operator Error
```

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:14
Your choise:14
/*
 *Function Name:LevelOrderTraverse
 *Module:Data structures
 *Parameter:BinaryTreePos&T, status(*visit) (BiTPos&node)
 *Return:status
 *Use:Travel the BinaryTree in level_order
 *The BiTree is null
 *Operator Error
```

12. 获得树中不存在的节点

```
Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:6
Your choise:6
/*
 *Function Name:LocateNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:BiTPos
 *Use:get the node with key e in the BiTree
 */
*Input the Key:10
*LevelOrderTraverse Success
*Not Found
*Operator Error
```

13. 为树中不存在的节点设置值

```
Windows PowerShell

1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:7
Your choise:7
/*
 *Function Name:Assign
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e, ElemtType value
 *Return:int
 *Use:find the node with key e in the BiTree
 *and change the value
*/
*Input the Key:10
*Input the value:
10
*LevelOrderTraverse Success
*Not Found
*Operator Error
```

14. 获得树中不存在的节点的兄弟节点

```
Windows PowerShell

1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1. dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:8
Your choise:8
/*
 *Function Name:GetSibling
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:BiTPos
 *Use:find the node with key e in BiTree
*/
*if node have lchild, return, else if have rchild
*/
*return, else return null
*/
*Input the Key:10
*LevelOrderTraverse Success
*Not Found
*Operator Error
```

15. 在树中不存在的节点位置插入节点

```
Windows PowerShell
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:9
Your choise:9
/*
 *Function Name:InsertNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e, int LR, BiTPos&c
 *Return:status
 *Use:insert a node
*/
*input the value of the node:10
*Input the Key:10
*LR:0 or 1: 0
*LevelOrderTraverse Success
*Not Found
*Operator Error
```

16. 删除树中不存在的节点

```
Windows PowerShell
-----Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign             8. GetSibling
9. InsertNode         10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir               -2. Show
-3. Open another menu -4. Find

请选择你的操作 [-4~17]:10
Your choise:10
/*
 *Function Name:DeleteNode
 *Module:Data structures
 *Parameter:BinaryTreePos&T, KeyType e
 *Return:status
 *Use:delete the node with key e
*/
*Input the key:10
*LevelOrderTraverse Success
*Not Found
*Operator Error
```

17. 加载不存在的二叉树文件

```

Windows PowerShell

Menu for Linear Table On Sequence Structure
-----index of present LinearList:1-----
1. CreateBiTree      2. DestroyBiTree
3. ClearBiTree       4. BiTreeEmpty
5. BiTreeDepth        6. LocateNode
7. Assign            8. GetSibling
9. InsertNode        10. DeleteNode
11. PreOrderTraverse 12. InOrderTraverse
13. PostOrderTraverse 14. LevelOrderTraverse
15. SaveBiTree        16. LoadBiTree
17. ChangeBiTree      0. Exit
-1.dir              -2. Show
-3.Open another menu -4. Find

请选择你的操作 [-4~17]:16
Your choise:16
/*
 *Function Name:LoadBiTree
 *Module:Data structures
 *Use:load the BiTree from a file
 */
*FileName:Tree_Error
*File open error

```

3.5 实验小结

这次实验和之前相比就略有复杂了，主要锻炼我们使用递归的能力。我的二叉树创建算法使用了前序遍历序列加上中序遍历序列，后序遍历序列加上中序遍历序列，前序遍历序列加上空节点，后序遍历序列加上空节点四种方式来唯一确定二叉树。

同时实验中用到了栈和队列，这两个数据结构我都自己编写好 ADT，包含结构体定义和基本运算操作，写在头文件里面，方便在测试文件中调用

自己在实验过程中遇到的问题主要出现在文件存储上，一开始在不知道用怎样的逻辑存储二叉树上困扰，考虑了较长时间才决定最终的实行方案

事实上本实验作为最基本的二叉树，主要是让我们熟练底层结构（倒不如说

是练习 C 语言)

本系统完整的实现了实现的课程要求的全部功能，实现了多二叉树管理和文件存储功能，系统健壮性良好，可以输出各种情况下的错误提示，并且在整个实验过程中避免了处处的内存泄漏，完整的达到了预期的效果。

4 基于邻接表的图实现

4.1 问题描述

实验基于邻接表存储结构的图，并实现图的基本运算。

实验中要求的图类型有：无向图，有向图，无向网，有向网，其中网是指带权的图。

要求构造一个具有菜单的功能演示系统。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，并给出适当的操作提示显示。演示系统可选择实现二叉树的文件形式保存。

演示系统可以实现多二叉树管理。

整个系统的设计模式如下：

4.1.1 图抽象数据类型

以下抽象数据类型源自参考文献[1]P156。

依据最小完备性和常用性相结合的原则，设计了图的数据对象和数据关系，并定义了图的初始化图、销毁图、查找顶点、获取邻接顶点等 13 种基本运算，具体数据和运算功能定义如下。

(1) 创建图：函数名称是 CreateCraph(G,V,VR)；初始条件是 V 是图的顶点集，VR 是图的关系集；操作结果是按 V 和 VR 的定义构造图 G。

注：要求图 G 中顶点关键字具有唯一性。后面各操作的实现，也都要满足一个图中关键字的唯一性，不再赘述；V 和 VR 对应的是图的逻辑定义形式，

比如 V 为顶点序列， VR 为关键字对的序列。不能将邻接矩阵等物理结构来代替 V 和 VR。

(2) 销毁图：函数名称是 DestroyGraph(G)；初始条件图 G 已存在；操作结果是销毁图 G。

(3) 查找顶点：函数名称是 LocateVex(G,u)；初始条件是图 G 存在， u 是和 G 中顶点关键字类型相同的给定值；操作结果是若 u 在图 G 中存在， 返回关键字为 u 的顶点位置信息，否则返回其它信息。

(4) 顶点赋值：函数名称是 PutVex (G,u,value)；初始条件是图 G 存在， u 是和 G 中顶点关键字类型相同的给定值；操作结果是对关键字为 u 的顶点赋值 value。

(5) 获得第一邻接点：函数名称是 FirstAdjVex(G, u)；初始条件是图 G 存在， u 是 G 中顶点的位序；操作结果是返回 u 对应顶点的第一个邻接顶点位置信息，如果 u 的顶点没有邻接顶点，否则返回其它表示“空”的信息。

(6) 获得下一邻接点：函数名称是 NextAdjVex(G, v, w)；初始条件是图 G 存在， v 和 w 是 G 中两个顶点的位序， v 对应 G 的一个顶点，w 对应 v 的邻接顶点；

操作结果是返回 v 的（相对于 w）下一个邻接顶点的位置信息，如果 w 是最后一个邻接顶点，返回其它表示“空”的信息。

(7) 插入顶点：函数名称是 InsertVex(G,v)；初始条件是图 G 存在，v 和 G 中的顶点具有相同特征；操作结果是在图 G 中增加新顶点 v。（在这里也保持顶点关键字的唯一性）

(8) 删除顶点：函数名称是 DeleteVex(G,v)；初始条件是图 G 存在，v 是 G 的一个顶点；操作结果是在图 G 中删除顶点 v 和与 v 相关的弧。

(9) 插入弧：函数名称是 InsertArc(G,v,w)；初始条件是图 G 存在，v、w 是 G 的顶点；操作结果是在图 G 中增加弧<v,w>，如果图 G 是无向图，还需要增加<w,v>。

(10) 删除弧：函数名称是 DeleteArc(G,v,w)；初始条件是图 G 存在，v、w 是 G 的顶点；操作结果是在图 G 中删除弧<v,w>，如果图 G 是无向图，还需要删除<w,v>。

(11) 深度优先搜索遍历：函数名称是 DFSTraverse(G,visit())；初始条件是图 G 存在；操作结果是图 G 进行深度优先搜索遍历，依次对图中的每一个顶点使

用函数 visit 访问一次，且仅访问一次。

(12) 广深度优先搜索遍历：函数名称是 BFSTraverse(G,visit())；初始条件是图 G 存在；操作结果是图 G 进行广度优先搜索遍历，依次对图中的每一个顶点使用函数 visit 访问一次，且仅访问一次。

4.1.2 多图的管理

我设计了一个结构体数组，将已存在的所有图用顺序存储方式存储，并对每个图赋予独特的 ID 来标识，可以通过 ID 来完成不同图之间的切换。

4.1.3 演示系统和文件存储的设计

使用了参考文件中的演示框架，该框架将完成函数调用所需实参值的准备和函数执行结果的实现，并给出适当的操作提示显示，整体以命令行呈现。

此外设计的数据文件实时存储，本地数据文件为 Graph_*, 每一个文件存储这一个图的数据。

4.2 系统设计

4.2.1 数据物理结构

1. 图的存储数据结构

```
typedef struct Graph
{
    int id;
    vexnode *first_vex;
    int vex_num;
    int arc_num;
}G,*Gpos;
```

2. 图的节点的存储数据结构

```
typedef struct VexNode
{
```

```

int index;
ElemType value;
arcnode *first_arc;
struct VexNode *nextvex;
}vexnode;

```

3.图的边的存储数据结构

```

typedef struct ArcNode
{
    int vex_index;
    struct ArcNode *nextarc;
    int weight;
}arcnode;

```

4.2.2 演示系统

演示系统包括用户操作界面和功能调用部分。

演示系统界面语言为英文,所有操作和提示语言均为英文。

用户操作界面输出可选的线性表操作,用户输入数字选择要进行的操作。在用户选择操作后,功能调用部分会显示函数的名称,参数,返回值和作用,系统提示用户输入参数。

功能调用部分将用户输入的有关信息传递给线性数据结构的操作函数进行调用,并对函数的返回值进行处理判断输出相应的提示信息。

4.2.3 图运算实现算法

1.status CreateGraph(GPos&G, int*vex, int*vex_arc);

功能:创建一个图。

算法实现:根据输入的 vex 数组和 vex_arc 数组执行插入函数创建结点和相应的边, 创建成功最后返回 OK,否则返回 ERROR,并 free 掉所有已经申请的空间。

时空效率分析:由于存在遍历 n^2 次来创建边, 算法的时间复杂度为 $O(n^2)$, 同理需要的空间也是这么大,因此空间复杂度为 $O(n^2)$ 。

2.status DestroyGraph(GPos&G);

功能:删除图。

算法实现:遍历并 free 掉所有的边和结点,最后 free 掉自身,返回 OK。

时空效率分析:算法的时间复杂度为 $O(m+n)$,为 m 个结点, n 条边,同理空间复杂度为 $O(m+n)$ 。

3.int LocateVex(GPos&G, int u);

功能:根据值查找结点。

算法实现:这里相当于链表的遍历查找操作,若查找到相应的结点,返回其值,否则返回 0。

时空效率分析:和链表的遍历查找操作一样,算法的时间复杂度为 $O(n)$,空间复杂度为 $O(1)$ 。

4.status PutVex(GPos&G, int u, ElemtType value);

功能:设置结点的值。

算法实现:查找操作,若找到相应的结点,那么其值设置为 value,返回 OK,否则返回 ERROR。

时空效率分析:时间复杂度为 $O(n)$,空间复杂度为 $O(1)$ 。

5.int FirstAdjVex(GPos&G, int u);

功能:获得第一条邻接的边。

算法实现:首先查找 u 结点是否存在,若存在,返回其 index 即可,否则返回 -1。

时空效率分析:算法的时间复杂度为查找所需的 $O(n)$,空间复杂度为 $O(1)$ 。

6.int NextAdjVex(GPos&G, int v, int w);

功能:获得指定边的下一条与节点连接的边。

算法实现:首先查找 v 结点是否存在,若存在,则寻找与这个结点连接的边 v-w, 若找到并且这条边的 nextarc 不为 NULL, 返回这个节点 p->nextarc->vex_index, 否则返回 -1。

时空效率分析:算法的时间复杂度为查找所需的 $O(n)$, 空间复杂度为 $O(1)$ 。

7.status InsertVex(GPos&G, vexnode*v);

功能:插入结点。

算法实现:为了保证广度优先搜索中序号的一致性,这里插入采用表尾插入。

时空效率分析:算法的时间复杂度为遍历表的 $O(n)$, 空间复杂度为 $O(1)$ 。

8.status DeleteVex(GPos&G, int v);

功能:删除结点。

算法实现:遍历查找 v 结点,若找到,删除其所有邻接的边,然后删除自身结点。

同时遍历所有边,删除指向该结点的边,最后返回 OK,否则返回 ERROR。

时空效率分析:算法的时间复杂度为 $O(m+n)$, 为 m 个结点条边,空间复杂度为 $O(1)$ 。

9.status InsertArc(GPos&G, int v, int w);

功能:插入边。

算法实现:首先搜索 v 结点和 w 是否存在,若不存在,返回 ERROR,否则在 v 结点的边链表中插入一条指向 w 的边,返回 OK。

时空效率分析:算法的时间复杂度为查找的 $O(n)$, 空间复杂度为 $O(1)$ 。

10.status DeleteArc(GPos&G, int v, int w);

功能:删除边。

算法实现:首先搜索 v 结点是否存在,若不存在,返回 ERROR,否则遍历 v 结点的边链表并删除这条边,若这条边不存在则返回 ERROR,否则删除成功返回 OK。

时空效率分析:算法的时间复杂度为查找的 $O(n)$,空间复杂度为 $O(1)$ 。

11.status DFSTraverse(GPos&G, void(*visit)(vexnode*vex));

功能:深度优先遍历图。

算法实现:首先设置一个数组 visit 作为标记数组,以标记结点是否被遍历过,数组的初始值均为 0,然后从第一个结点开始,对每个结点执行以下操作:

1.打印该结点的值,并且把 visit 值赋值为 1

2.若第一个邻接结点没有被访问过,则对其执行操作

3.若其下一个邻接结点没有被访问过,则对其执行操作

4.....

Fin.若其最后一个邻接结点没有被访问过,则对其执行操作

时空效率分析:算法的时间复杂度为 $O(m+n)$,为 m 个结点和 n 条边,空间复杂度为 $O(m)$ 。

12.status BFSTraverse(GPos&G, void(*visit)(vexnode*vex));

功能:广度优先遍历图。

算法实现:首先设置一个数组 visit 作为标记数组,以标记结点是否被遍历过,数组的初始值均为 0,然后使用一个辅助队列,对每个结点执行以下操作:

1.若该结点没有被访问过,则该结点进入队列 2, 若队列不为空,则队列首元素出列,打印该结点的值,并且把 visit 值赋值为 1

2.若该结点的第一个邻接结点没有被访问过,则进入队列尾

- 3.若该结点的下一个邻接结点没有被访问过,则进入队列尾
 - 4.....
- Fin.若该结点的最后一个邻接结点没有被访问过,则进入队列尾,跳转至步

骤 2

时空效率分析:算法的时间复杂度为 $O(m+n)$,为 m 个结点和 n 条边,空间复杂度为 $O(m)$ 。

4.2.4 多图管理实现算法

由于多图的管理采用结构体数组存储结构,只涉及到查找操作,所以时间复杂度是 $O(1)$,空间复杂度为 $O(1)$ 。

4.2.5 文件储存实现算法

将一个图存储为一个文件的时候,将图的点集和边集存在两个数组里面,一个存点,一个存边,然后将数组中的元素依次存进二进制文件中,这个文件里面就存储了一个图。然后读取文件的时候,将文件里面的元素按顺序读取进两个数组中,然后用这两个数组来创建一个新图。时间复杂度为 $O(n)$,空间复杂度为 $O(1)$ 。

4.3 系统实现

4.3.1 实验环境

实验环境为 xUbuntu 19.10, 编译器为 gcc/g++ (Ubuntu 9.2.1-9ubuntu2) 9.2.1 20191008。代码采用 Linux 系统自带文本编辑器 vim 编写。由自己写的 Makefile 来完成编译。

文件说明:

* GraphADT.h: 图库头文件

* Graph.cpp: 测试文件

* GraphADT.o: .o 文件

* Graph.o: .o 文件

* Graph.out: 可执行文件

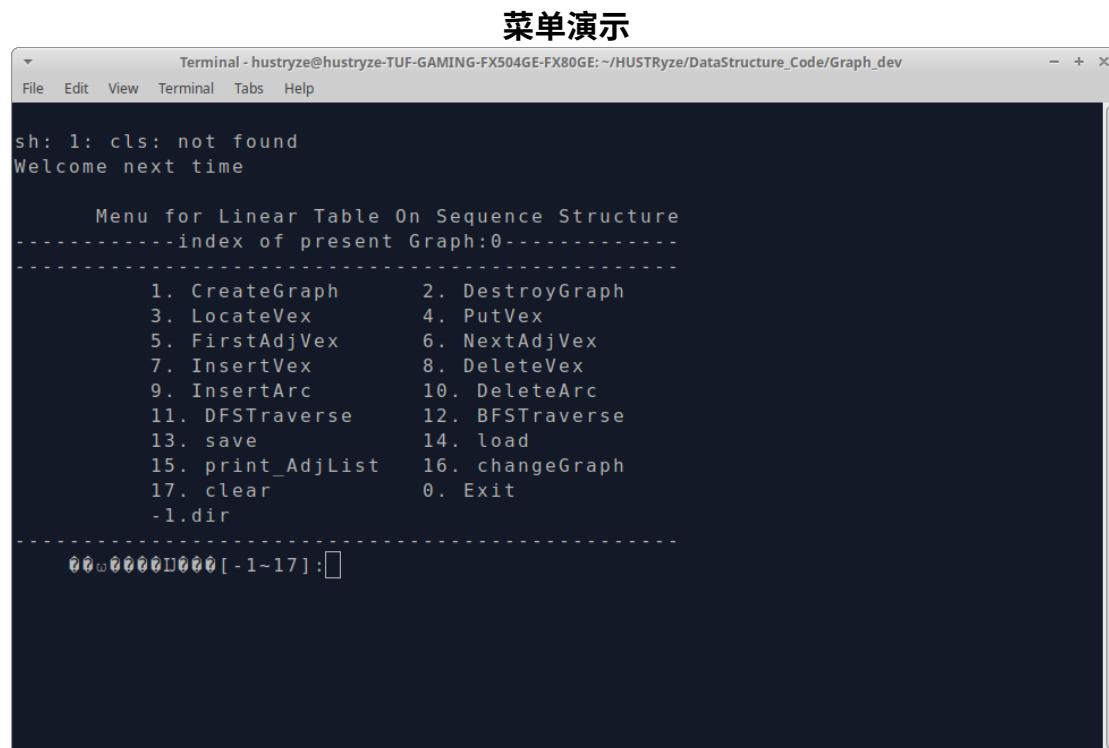
* Graph_*: 存储好的图文件

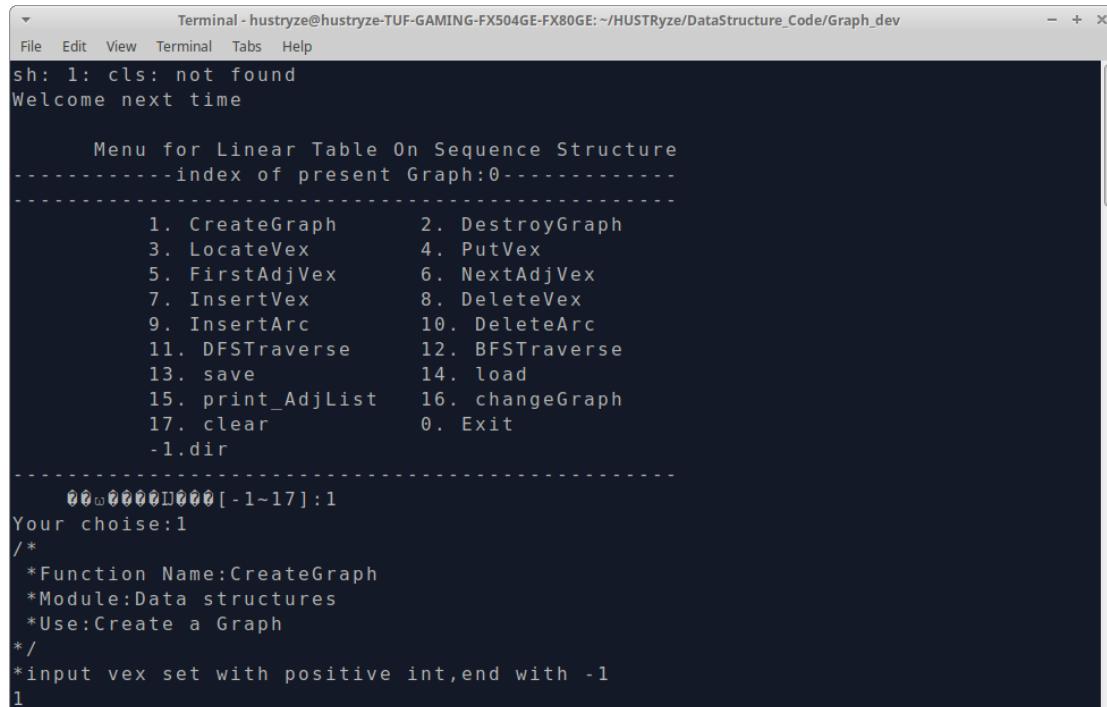
* Makefile：自主编写的 Makefile 文件

4.3.2 代码亮点

所有的代码采用 Google C/C++ 标准代码规范, 函数库所有注释采用英文, 符合生产规范。错误检查和提示全面, 根据不同的错误会有不同的提示信息。由于在 Linux 环境下编程, 所有的 API 接口命名改用 unix 环境编程的标准短横线命名模式。自主编写 Makefile 文件, 使得编译特别便捷与快速。

4.3.3 操作演示



(1) 创建一个图


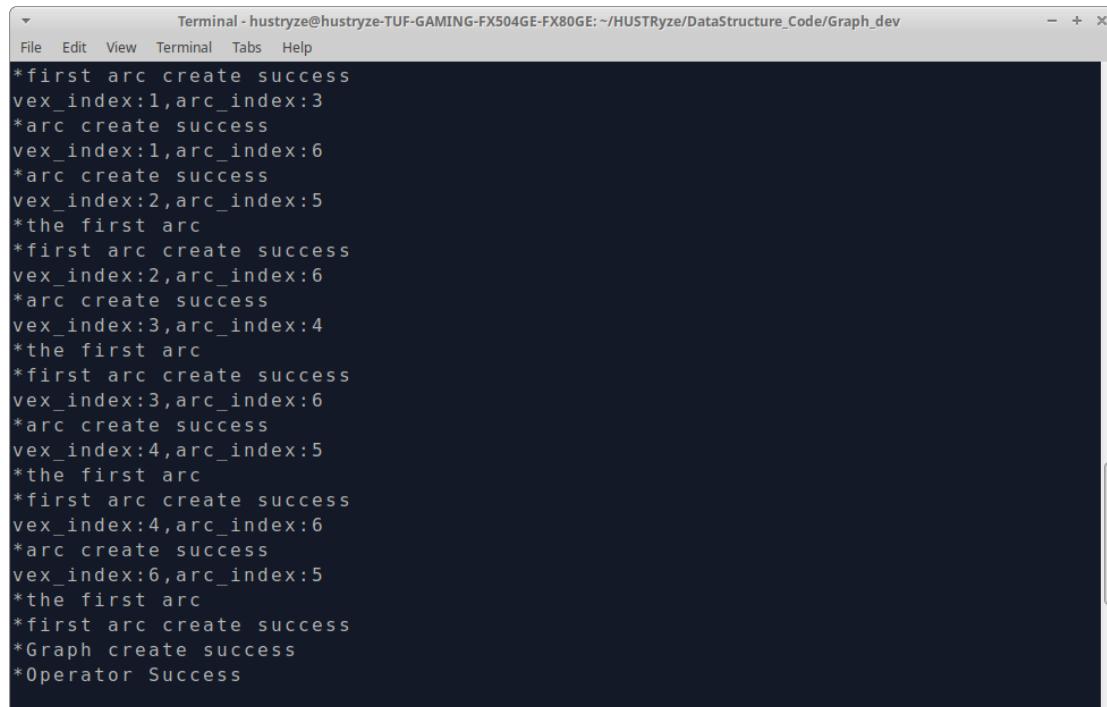
```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
sh: 1: cls: not found
Welcome next time

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

Your choise:1
/*
 *Function Name:CreateGraph
 *Module:Data structures
 *Use:Create a Graph
 */
*input vex set with positive int,end with -1
1

```



```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
*first arc create success
vex_index:1,arc_index:3
*arc create success
vex_index:1,arc_index:6
*arc create success
vex_index:2,arc_index:5
*the first arc
*first arc create success
vex_index:2,arc_index:6
*arc create success
vex_index:3,arc_index:4
*the first arc
*first arc create success
vex_index:3,arc_index:6
*arc create success
vex_index:4,arc_index:5
*the first arc
*first arc create success
vex_index:4,arc_index:6
*arc create success
vex_index:6,arc_index:5
*the first arc
*first arc create success
*Graph create success
*Operator Success

```

(2) 打印邻接表

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
      3. LocateVex      4. PutVex
      5. FirstAdjVex    6. NextAdjVex
      7. InsertVex      8. DeleteVex
      9. InsertArc       10. DeleteArc
     11. DFSTraverse    12. BFSTraverse
     13. save           14. load
     15. print_AdjList  16. changeGraph
     17. clear          0. Exit
     -1.dir

-----
    00w0000D000[-1~17]:15
Your choise:15
/*
 *Function Name:print_AdjList
 *Module:Data structures
 *Use:print the AdjList
*/
1->2->3->6->
2->5->6->
3->4->6->
4->5->6->
5->
6->5->
*print AdjList success
*Operator Success

```

(3) 删除这个图

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
      1. CreateGraph      2. DestroyGraph
      3. LocateVex        4. PutVex
      5. FirstAdjVex      6. NextAdjVex
      7. InsertVex         8. DeleteVex
      9. InsertArc        10. DeleteArc
     11. DFSTraverse     12. BFSTraverse
     13. save            14. load
     15. print_AdjList   16. changeGraph
     17. clear           0. Exit
     -1.dir

-----
    00w0000D000[-1~17]:2
Your choise:2
/*
 *Function Name:DestroyGraph
 *Module:Data structures
 *Use:destroy the Graph
*/
*Destroy success
*Operator Success

```

(4) 打印邻接表，显示错误

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

@@@@@@I@[-1~17]:15
Your choise:15
/*
 *Function Name:print_AdjList
 *Module:Data structures
 *Use:print the AdjList
*/
*The Graph is NULL
*Operator Error

```

(5) 重新创建这个图

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
sh: 1: cls: not found
Welcome next time

Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

@@@@@@I@[-1~17]:1
Your choise:1
/*
 *Function Name:CreateGraph
 *Module:Data structures
 *Use>Create a Graph
*/
*input vex set with positive int,end with -1
1

```

```
*first arc create success
vex_index:1,arc_index:3
*arc create success
vex_index:1,arc_index:6
*arc create success
vex_index:2,arc_index:5
*the first arc
*first arc create success
vex_index:2,arc_index:6
*arc create success
vex_index:3,arc_index:4
*the first arc
*first arc create success
vex_index:3,arc_index:6
*arc create success
vex_index:4,arc_index:5
*the first arc
*first arc create success
vex_index:4,arc_index:6
*arc create success
vex_index:6,arc_index:5
*the first arc
*first arc create success
*Graph create success
*Operator Success
```

(6) 找到关键字为 3 的节点

```
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

0000000000[-1~17]:3
Your choise:3
/*
 *Function Name:LocateVex
 *Module:Data structures
 *Use:locate the vex node
 */
*input u:3
*Found
*Location:2
[]
```

(7) 给关键字为 3 的节点赋值

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

000000000000[-1~17]:4
Your choise:4
/*
 *Function Name:PutVex
 *Module:Data structures
 *Use:Assign the value of the vex node
 */
*input u:3
*input value:3
*Found
*Assign success:3
*Operator Success
[]
```

(8) 获得关键字为 3 的节点的第一个邻接点

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

000000000000[-1~17]:5
Your choise:5
/*
 *Function Name:FirstAdjVex
 *Module:Data structures
 *Use:return first adjvex of the vex node
 */
*input u:3
*Found u
*Location:3
[]
```

(9) 获得 (3,4) 的下一个邻接点

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

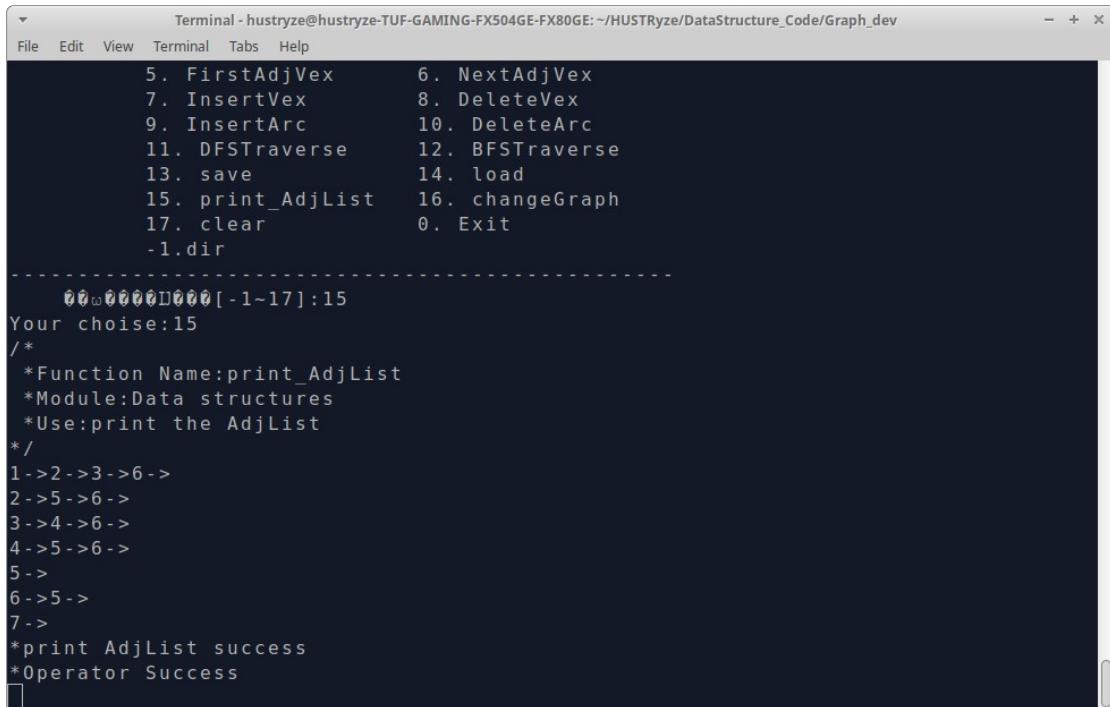
000000000000[-1~17]:6
Your choise:6
/*
 *Function Name:NextAdjVex
 *Module:Data structures
 *Use:get the next adjvex
 */
*input v:3
*input w:4
*Found v
*Found w
*Location:5
□
```

(10) 插入一个关键字为 7,值为 7 的节点

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

000000000000[-1~17]:7
Your choise:7
/*
 *Function Name:InsertVex
 *Module:Data structures
 *Use:Insert a vex node
 *input index:7
 *input value:7
 *Insert vex success
 *Operatoe Success
□
```

(11) 打印邻接表

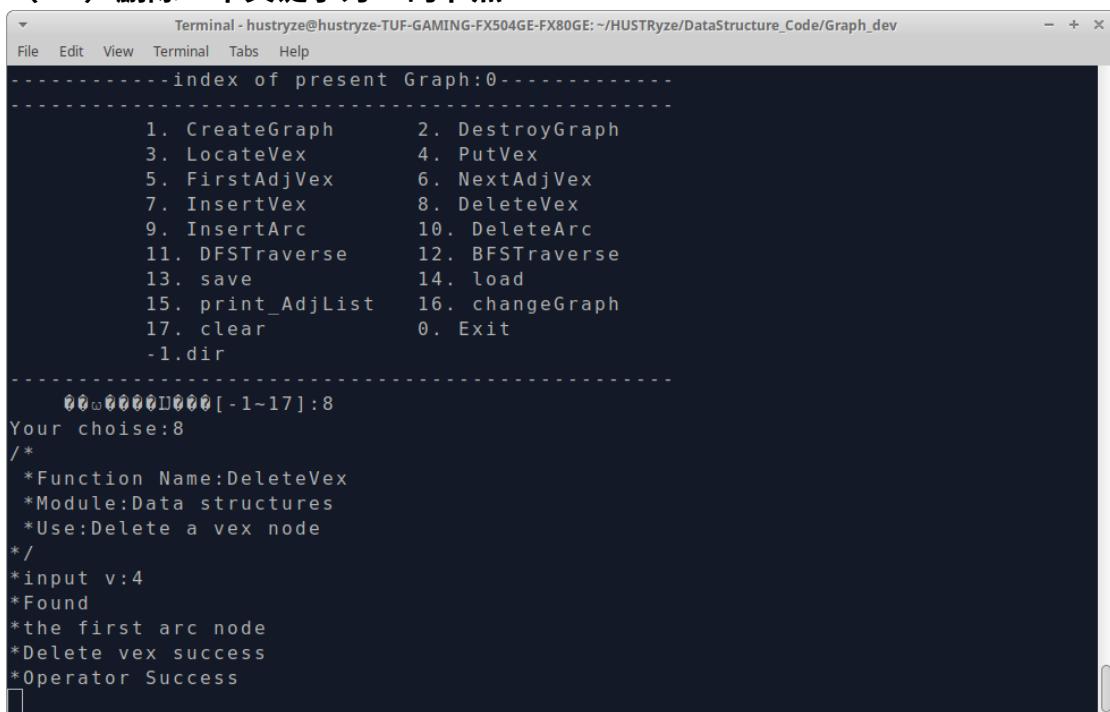


```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
      5. FirstAdjVex      6. NextAdjVex
      7. InsertVex        8. DeleteVex
      9. InsertArc         10. DeleteArc
     11. DFSTraverse       12. BFSTraverse
     13. save             14. load
     15. print_AdjList    16. changeGraph
     17. clear             0. Exit
     -1.dir
-----
@0@0@0@0@0@0@0@[-1~17]:15
Your choise:15
/*
 *Function Name:print_AdjList
 *Module:Data structures
 *Use:print the AdjList
*/
1->2->3->6->
2->5->6->
3->4->6->
4->5->6->
5->
6->5->
7->
*print AdjList success
*Operator Success

```

(12) 删除一个关键字为 4 的节点



```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
-----index of present Graph:0-----
      1. CreateGraph      2. DestroyGraph
      3. LocateVex         4. PutVex
      5. FirstAdjVex       6. NextAdjVex
      7. InsertVex         8. DeleteVex
      9. InsertArc          10. DeleteArc
     11. DFSTraverse        12. BFSTraverse
     13. save              14. load
     15. print_AdjList      16. changeGraph
     17. clear              0. Exit
     -1.dir
-----
@0@0@0@0@0@0@0@[-1~17]:8
Your choise:8
/*
 *Function Name:DeleteVex
 *Module:Data structures
 *Use:Delete a vex node
*/
*input v:4
*Found
*the first arc node
*Delete vex success
*Operator Success

```

(13) 打印邻接表

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
3. LocateVex      4. PutVex
5. FirstAdjVex   6. NextAdjVex
7. InsertVex     8. DeleteVex
9. InsertArc     10. DeleteArc
11. DFSTraverse  12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear         0. Exit
-1.dir

00w0000D000 [-1~17]:15
Your choise:15
/*
 *Function Name:print_AdjList
 *Module:Data structures
 *Use:print the AdjList
*/
1->2->3->6->
2->5->6->
3->6->
5->
6->5->
7->
*print AdjList success
*Operator Success

```

(14) 插入一条从 6 到 7 的边

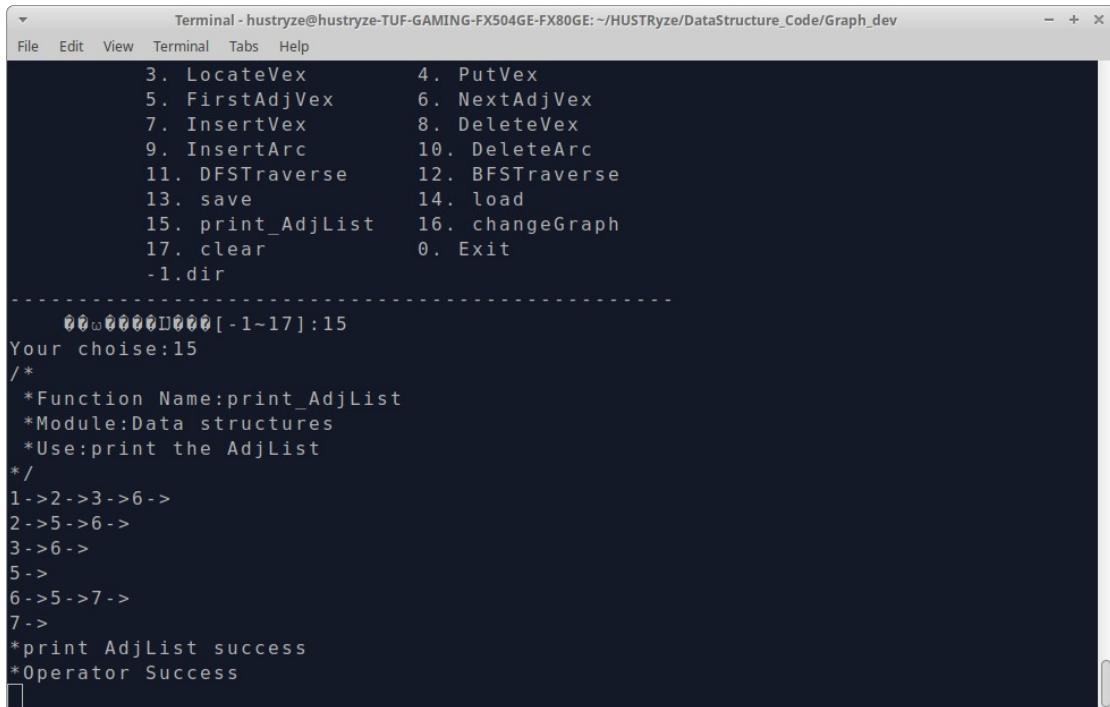
```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
1. CreateGraph    2. DestroyGraph
3. LocateVex      4. PutVex
5. FirstAdjVex   6. NextAdjVex
7. InsertVex     8. DeleteVex
9. InsertArc     10. DeleteArc
11. DFSTraverse  12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear         0. Exit
-1.dir

00w0000D000 [-1~17]:9
Your choise:9
/*
 *Function Name:InsertArc
 *Module:Data structures
 *Use:Insert an arc node
*/
*input v:6
*input w:7
*Found w
*Found v
*Insert arc success
*Operator Success

```

(15) 打印邻接表



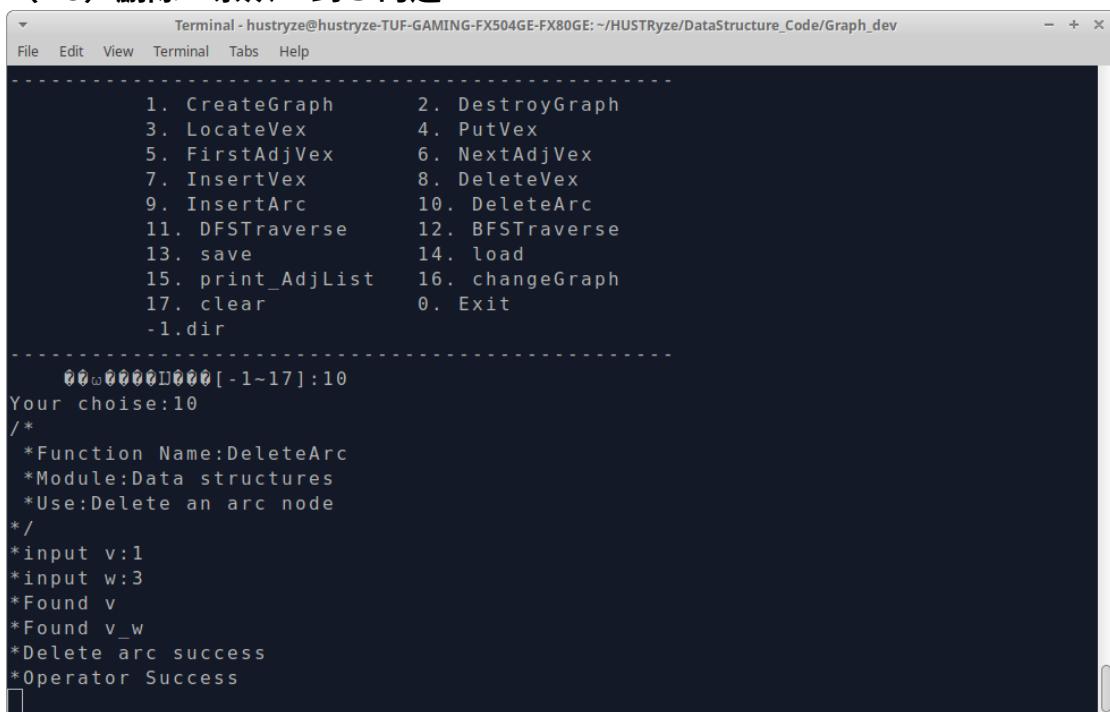
```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
3. LocateVex      4. PutVex
5. FirstAdjVex   6. NextAdjVex
7. InsertVex     8. DeleteVex
9. InsertArc     10. DeleteArc
11. DFSTraverse  12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear          0. Exit
-1.dir

00w0000D000 [-1~17]:15
Your choise:15
/*
 *Function Name:print_AdjList
 *Module:Data structures
 *Use:print the AdjList
*/
1->2->3->6->
2->5->6->
3->6->
5->
6->5->7->
7->
*print AdjList success
*Operator Success

```

(16) 删除一条从 1 到 3 的边



```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
1. CreateGraph    2. DestroyGraph
3. LocateVex      4. PutVex
5. FirstAdjVex   6. NextAdjVex
7. InsertVex     8. DeleteVex
9. InsertArc     10. DeleteArc
11. DFSTraverse  12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear          0. Exit
-1.dir

00w0000D000 [-1~17]:10
Your choise:10
/*
 *Function Name:DeleteArc
 *Module:Data structures
 *Use:Delete an arc node
*/
*input v:1
*input w:3
*Found v
*Found v_w
*Delete arc success
*Operator Success

```

(17) 打印邻接表

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
3. LocateVex      4. PutVex
5. FirstAdjVex    6. NextAdjVex
7. InsertVex      8. DeleteVex
9. InsertArc       10. DeleteArc
11. DFSTraverse   12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear          0. Exit
-1.dir

00w0000D000[-1~17]:15
Your choise:15
/*
 *Function Name:print_AdjList
 *Module:Data structures
 *Use:print the AdjList
*/
1->2->6->
2->5->6->
3->6->
5->
6->5->7->
7->
*print AdjList success
*Operator Success

```

(18) 深度遍历

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
3. LocateVex      4. PutVex
5. FirstAdjVex    6. NextAdjVex
7. InsertVex      8. DeleteVex
9. InsertArc       10. DeleteArc
11. DFSTraverse   12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear          0. Exit
-1.dir

00w0000D000[-1~17]:11
Your choise:11
/*
 *Function Name:DFSTraverse
 *Module:Data structures
 *Use:Travel the Graph in DFS order
*/
index:1 value:0
index:2 value:0
index:5 value:0
index:6 value:0
index:7 value:7
index:3 value:3
*DFS Traverse success
*Operator Success

```

(19) 广度遍历

```

Terminal - hustrye@hustrye-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
3. LocateVex      4. PutVex
5. FirstAdjVex    6. NextAdjVex
7. InsertVex      8. DeleteVex
9. InsertArc      10. DeleteArc
11. DFSTraverse   12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear          0. Exit
-1.dir

-----
@@@0000D000[-1~17]:12
Your choise:12
/*
 *Function Name:BFSTraverse
 *Module:Data structures
 *Use:Travle the Graph in BFS order
*/
index:1 value:0
index:2 value:0
index:6 value:0
index:5 value:0
index:7 value:7
index:3 value:3
*BFS Traverse success
*Operator Success

```

(20) 保存这个图为“Graph_03”

```

Terminal - hustrye@hustrye-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph     2. DestroyGraph
3. LocateVex       4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex       8. DeleteVex
9. InsertArc       10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save           14. load
15. print_AdjList  16. changeGraph
17. clear           0. Exit
-1.dir

-----
@@@0000D000[-1~17]:13
Your choise:13
/*
 *Function Name:save
 *Module:Data structures
 *Use:save this Graph as a file
 *file name:Graph_03
1 2 3 5 6 7
1,6 1,6 2,6 2,6 3,6 6,6      *write success

```

(21) 查看本地文件，发现已经保存成功

```

Terminal - hustrye@hustrye-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
*Use:save this Graph as a file
*file name:Graph_03
1      2      3      5      6      7
1,6    1,6    2,6    2,6    3,6    6,6    6,6    *write success

sh: 1: cls: not found
Welcome next time

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir
-----
@@@000000000[-1~17]:-1
Graph_01  Graph_03  GraphADT.h  Graph.cpp  Graph.out  Makefile.win
Graph_02  Graph_04  GraphADT.o  Graph.o    Makefile

```

(22) 切换到索引为 1 的另一个图

```

Terminal - hustrye@hustrye-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir
-----
@@@000000000[-1~17]:16
Your choise:16
/*
 *Function Name:ChangeGraph
 *Module:Data structures
 *Use:change the Graph
*/
index:1
*change success

```

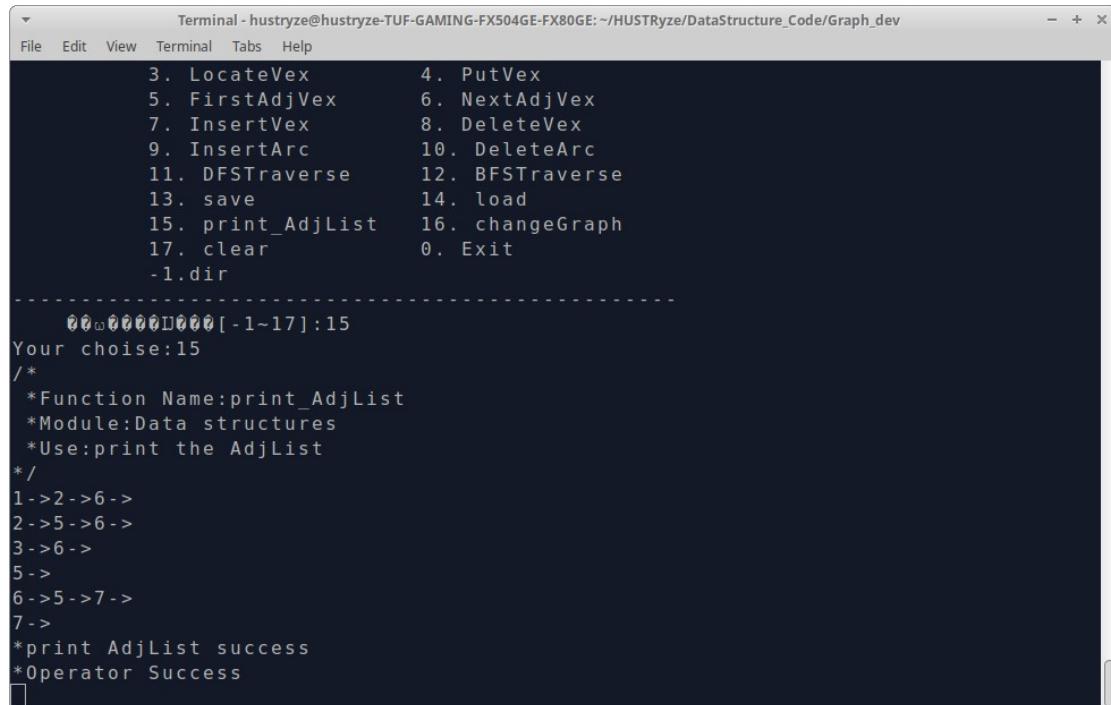
(23) 加载刚才保存的图文件

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
-1.dir
-----
@0w@0000D000[-1~17]:14
Your choise:14
/*
 *Function Name:Load
 *Module:Data structures
 *Use:Load a Graph from a file
*/
*FileName:Graph_03
*load vex success
*load arc success
1
2
3
5
6
7

1,2      1,6      2,5      2,6      3,6      6,5      6,7
*vex num:6
*arc num:7
index:1,value:0
index:2,value:0
index:3,value:0
index:5,value:0
```

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
index:3,value:0
index:5,value:0
index:6,value:0
index:7,value:0
*vex create success
vex_index:1,arc_index:2
*the first arc
*first arc create success
vex_index:1,arc_index:6
*arc create success
vex_index:2,arc_index:5
*the first arc
*first arc create success
vex_index:2,arc_index:6
*arc create success
vex_index:3,arc_index:6
*the first arc
*first arc create success
vex_index:6,arc_index:5
*the first arc
*first arc create success
vex_index:6,arc_index:7
*arc create success
*Graph create success
*Read Success
[]
```

(24) 打印邻接表



```

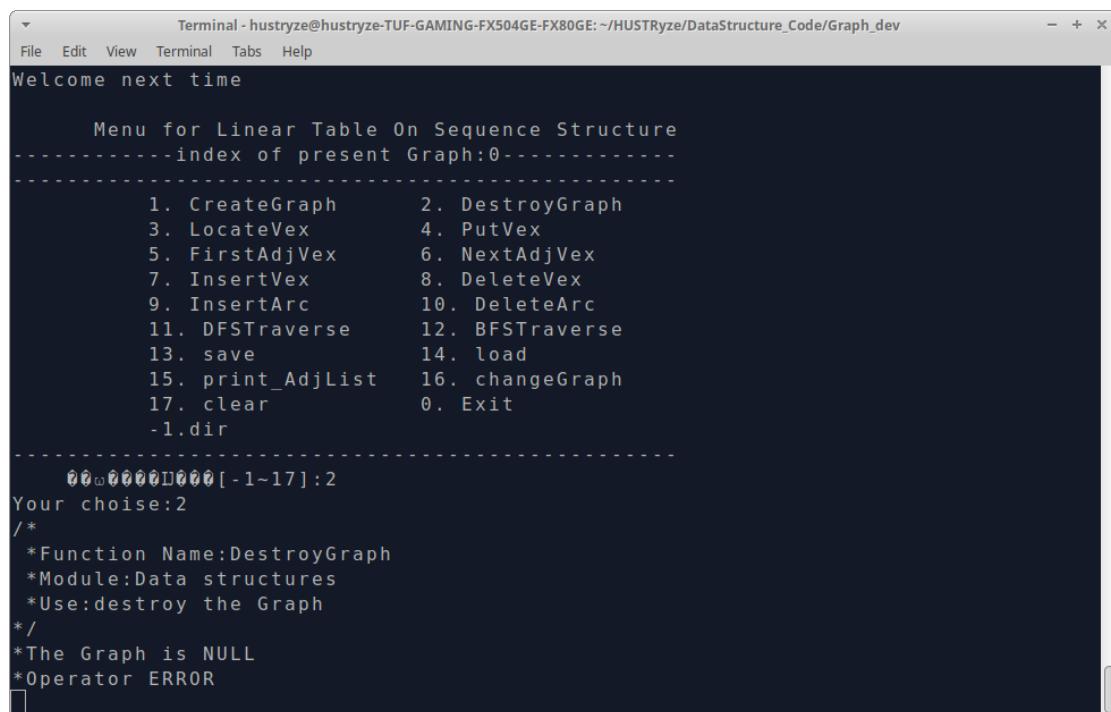
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
3. LocateVex      4. PutVex
5. FirstAdjVex   6. NextAdjVex
7. InsertVex     8. DeleteVex
9. InsertArc     10. DeleteArc
11. DFSTraverse  12. BFSTraverse
13. save          14. load
15. print_AdjList 16. changeGraph
17. clear         0. Exit
-1.dir

Your choise:15
/*
 *Function Name:print_AdjList
 *Module:Data structures
 *Use:print the AdjList
*/
1->2->6->
2->5->6->
3->6->
5->
6->5->7->
7->
*print AdjList success
*Operator Success

```

4.4 系统测试

(1) 删除不存在的图



```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList  16. changeGraph
17. clear           0. Exit
-1.dir

Your choise:2
/*
 *Function Name:DestroyGraph
 *Module:Data structures
 *Use:destroy the Graph
*/
/*The Graph is NULL
*Operator ERROR

```

(2) 对不存在的图进行操作

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

-----[00w0000I000[-1~17]:3
Your choise:3
/*
 *Function Name:LocateVex
 *Module:Data structures
 *Use:locate the vex node
*/
*input u:1
*The Graph is NULL
*Location:-1
[ ]

```

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

-----[00w0000I000[-1~17]:4
Your choise:4
/*
 *Function Name:PutVex
 *Module:Data structures
 *Use:Assign the value of the vex node
*/
*input u:1
*input value:1
*The Graph is NULL
*Operator Error
[ ]

```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex      6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse     12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

00w0000I000[-1~17]:5
Your choise:5
/*
 *Function Name:FirstAdjVex
 *Module:Data structures
 *Use:return first adjvex of the vex node
*/
*input u:1
*The Graph is NULL
*Location:-1
[]
```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex      6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse     12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

00w0000I000[-1~17]:6
Your choise:6
/*
 *Function Name:NextAdjVex
 *Module:Data structures
 *Use:get the next adjvex
*/
*input v:1
*input w:1
*The Graph is NULL
*Location:-1
[]
```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

00w0000I000[-1~17]:7
Your choise:7
/*
 *Function Name:InsertVex
 *Module:Data structures
 *Use:Insert a vex node
 *input index:8
 *input value:8
 *The Graph is NULL
 *Operator Error
[]


```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

00w0000I000[-1~17]:8
Your choise:8
/*
 *Function Name:DeleteVex
 *Module:Data structures
 *Use>Delete a vex node
 */
*input v:1
 *The Graph is NULL
 *Operator Error
[]


```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

    00w0000I000[-1~17]:9
Your choise:9
/*
 *Function Name:InsertArc
 *Module:Data structures
 *Use:Insert an arc node
*/
*input v:1
*input w:1
*The Graph is NULL
*Operator Error

```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

    00w0000I000[-1~17]:10
Your choise:10
/*
 *Function Name:DeleteArc
 *Module:Data structures
 *Use:Delete an arc node
*/
*input v:1
*input w:1
*The Graph is NULL
*Operator Error

```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir
-----
@@@@@@I@[-1~17]:11
Your choise:11
/*
 *Function Name:DFSTraverse
 *Module:Data structures
 *Use:Travel the Graph in DFS order
 */
*The Graph is NULL
*Operator Error

```

```

Terminal - hstryze@hstryze-TUF-GAMING-FX504GE-FX80GE:~/HUSTRye/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

      Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc        10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir
-----
@@@@@@I@[-1~17]:12
Your choise:12
/*
 *Function Name:BFSTraverse
 *Module:Data structures
 *Use:Travle the Graph in BFS order
 */
*The Graph is NULL
*Operator Error

```

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

        Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir
-----
@@@00000I000[-1~17]:14
Your choise:14
/*
 *Function Name:Load
 *Module:Data structures
 *Use:Load a Graph from a file
*/
*FileName:Error
*File open error
[]
```

(3) 寻找不存在的节点

```
Terminal - hustryze@hstryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir
-----
@@@00000I000[-1~17]:3
Your choise:3
/*
 *Function Name:LocateVex
 *Module:Data structures
 *Use:locate the vex node
*/
*input u:10
*Not Found
*Location:-1
[]
```

(4) 给不存在的节点赋值

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HISTRYze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex      6. NextAdjVex
7. InsertVex         8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse     12. BFSTraverse
13. save             14. load
15. print_AdjList   16. changeGraph
17. clear             0. Exit
-1.dir

0000000000[-1~17]:4
Your choise:4
/*
*Function Name:PutVex
*Module:Data structures
*Use:Assign the value of the vex node
*/
*input u:9
*input value:9
*Not Found
*Operator Error

```

(5) 找到不存在节点的第一个邻接点

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HISTRYze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex      6. NextAdjVex
7. InsertVex         8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse     12. BFSTraverse
13. save             14. load
15. print_AdjList   16. changeGraph
17. clear             0. Exit
-1.dir

0000000000[-1~17]:5
Your choise:5
/*
*Function Name:FirstAdjVex
*Module:Data structures
*Use:return first adjvex of the vex node
*/
*input u:10
*Not Found u
*Location:-1

```

(6) 输入不存在的邻接点

```
Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

    00w0000I000[-1~17]:6
Your choise:6
/*
 *Function Name:NextAdjVex
 *Module:Data structures
 *Use:get the next adjvex
*/
*input v:1
*input w:7
*Found v
*Not Found w
*Location:-1
[]
```

(7) 删除不存在的节点

```
Terminal - hustryze@hstryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

    00w0000I000[-1~17]:8
Your choise:8
/*
 *Function Name:DeleteVex
 *Module:Data structures
 *Use:Delete a vex node
*/
*input v:10
*Not Found
*Operator Error
[]
```

(8) 插入边有有节点不存在

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

00w0000I000[-1~17]:9
Your choise:9
/*
 *Function Name:InsertArc
 *Module:Data structures
 *Use:Insert an arc node
*/
*input v:1
*input w:10
*the node w dont exist
*Operator Error

```

(9) 删除不存在的边

```

Terminal - hustryze@hstryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
-----index of present Graph:0-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir

00w0000I000[-1~17]:10
Your choise:10
/*
 *Function Name:DeleteArc
 *Module:Data structures
 *Use>Delete an arc node
*/
*input v:1
*input w:10
*Found v
*dont have v_w
*Operator Error

```

(10) 加载不存在的文件

```

Terminal - hustryze@hustryze-TUF-GAMING-FX504GE-FX80GE: ~/HUSTRyze/DataStructure_Code/Graph_dev
File Edit View Terminal Tabs Help
Welcome next time

Menu for Linear Table On Sequence Structure
-----index of present Graph:0-----
-----
1. CreateGraph      2. DestroyGraph
3. LocateVex        4. PutVex
5. FirstAdjVex     6. NextAdjVex
7. InsertVex        8. DeleteVex
9. InsertArc         10. DeleteArc
11. DFSTraverse    12. BFSTraverse
13. save            14. load
15. print_AdjList   16. changeGraph
17. clear            0. Exit
-1.dir
-----
@@@w@@@II@@@[-1~17]:14
Your choise:14
/*
 *Function Name:Load
 *Module:Data structures
 *Use:Load a Graph from a file
 */
*FileName:Error
*File open error

```

4.5 实验小结

感觉图的实验写起来比二叉树要舒服的多，我更改了课本上对于图结构体的设计，用链式存储结构不仅使得节点个数的限制没有了，而且在许多操作上都更加的简洁与快速。

这次设计了单独的存储结构，减少了文件存储量和文件操作的复杂程度和代码量。

整个实验在 linux 环境下编程，所有的代码采用 Google C/C++ 标准代码规范，函数库所有注释采用英文，符合生产规范。错误检查和提示全面，根据不同的错误会有不同的提示信息。由于在 Linux 环境下编程，所有的 API 接口命名改用 unix 环境编程的标准短横线命名模式。自主编写 Makefile 文件，使自己对整个程序

的编译过程更加熟悉，使自己脱离 IDE，走上了自立更生编译文件的道路。

本系统完整的实现了实现的课程要求的全部功能,实现了多图管理和文件存储功能,系统健壮性良好,可以输出各种情况下的错误提示,并且在整个实验过程中避免了处处的内存泄漏,完整的达到了预期的效果。

整个四次数据结构实验结束了,个人感觉,其实还算都是些比较简单的东西,主要考察的是结构实现,对于算法问题到并没有怎么涉及,写起来也不是需要特别长的时间。

一点遗憾就是在最后才在 linux 系统上编程，其实在 linux 上编程比在 windows 编程要方便，快捷，最重要的是简洁。只不过在笔记本上装 linux 双系统花了不少时间。

期待课设,希望课设能出个有意思的题目。

参考文献

- [1] 严蔚敏等. 数据结构(C 语言版). 清华大学出版社
- [2] Larry Nyhoff. ADTs, Data Structures, and Problem Solving with C++. Second Edition, Calvin College, 2005
- [3] 殷立峰. Qt C++跨平台图形界面程序设计基础. 清华大学出版社, 2014:192~197
- [4] 严蔚敏等. 数据结构题集(C 语言版). 清华大学出版社

附录 A 基于顺序存储结构线性表实现的源程序

```

status ListInsert(SqListP L,int i,ElemType e)
{
    //在顺序线性表 L 中第 i 个位置之前插入新的元素 e
    if(L->elem==NULL)//判断 L 是否存在
    {
        printf("线性表不存在\n");
        return ERROR;
    }
    //i 的合法值为 1<=i<=L->length+1
    if(i<1||i>L->length+1)//i 值不合法
    {
        printf("OVERFLOW");
        return ERROR;
    }
    if(L->length>=L->listsize)
    {
        //当前存储空间已经满，增加分配
        ElemtType* newbase=(ElemtType*)realloc(L->elem,(L-
>listsize+LISTINCREMENT)*sizeof(ElemtType));
        if(!newbase)exit(OVERFLOW);//分配内存失败
        L->elem=newbase;//新基址
        L->listsize+=LISTINCREMENT;//增加存储容量
    }
    ElemtType* q=&(L->elem[i-1]);//q 为插入位置
    for(ElemtType*p=&(L->elem[L->length-1]);p>=q;--p)
        *(p+1)=*p;//插入位置及之后的元素右移
    *q=e;//插入 e
    ++L->length;//表长增加 1
    return OK;
}

}//ListInsert

```

```
status ListTraverse(SqListP L,void(*visit)(ElemType*e))  
{  
    //依次对 L 的每个数据元素调用函数 visit()  
  
    if(L->elem==NULL)//判断 L 是否存在  
    {  
        printf("线性表不存在\n");  
        return ERROR;  
    }  
  
    int i=0;//从表头开始遍历  
  
    while(i<(L->length))  
    {  
        visit(&(L->elem[i]));//对每个数据元素调用 visit  
        i++;  
    }  
    printf("\n");  
    return OK;  
}  
  
//写文件  
printf("input file name:");  
scanf("%s",writefilename);  
if((fp=fopen(writefilename,"wb"))==NULL)  
{  
    printf("File open errore\n ");  
    getchar();  
    getchar();  
    break;  
}  
fwrite(L->elem,sizeof(ElemType),L->length,fp);  
fclose(fp);
```

```
Printf("文件写入成功");

//读文件

printf("input file name:");
scanf("%s",readfilename);
L->length=0;
if((fp=fopen(readfilename,"rb"))==NULL)
{
    printf("File open error\n ");
    getchar();
    getchar();
    break;
}
while(fread(&L->elem[L->length],sizeof(ElemType),1,fp))
    L->length++;
fclose(fp);

printf("文件读取成功");
```

附录 B 基于链式存储结构线性表实现的源程序

```
status ListTraverse(LinkList &L, void(*visit)(ElemType e))
{
    //Travel the LinearList
    //Error Alarm
    if (L == NULL)
    {
        printf("*The LinearList do not exist\n");
        return ERROR;
    }
    LinkList pos = L->next;//initialize
    while (pos)
    {
        visit(pos->data);//call visit()
        pos = pos->next;
    }
    printf("\n");
    return OK;
}//ListTraverse
```

```
status ListSort(LinkList &L)
{
    //sort the LinearList
    //Error Alarm
    if (L == NULL)
    {
        printf("*The LinearList do not exist\n");
        return ERROR;
```

```
}

int i, j;
LinkList temp;
for (i = 0; i < L->data-1; i++)
{
    for (j = 0, temp = L->next; j < L->data - i-1; j++)
    {
        if (temp->data > temp->next->data)
        {
            ElemtType x = temp->data;
            temp->data = temp->next->data;
            temp->next->data = x;
        }
        temp = temp->next;
    }
}

return OK;
}//ListSort

status ListMerger(LinkList &La, LinkList &Lb, LinkList &Lc)
{
    //Merge two LinearList as a new one
    //Error Alarm
    if (La== NULL||Lb==NULL||Lc==NULL)
    {
        printf("*The LinearList do not exist\n");
        return ERROR;
    }
}
```

```
Lc->data = La->data + Lb->data;//compute length
LinkList temp,pos,semp_a,semp_b;
pos = Lc;
semp_a = La->next;
semp_b = Lb->next;
for (int i = 0; i < La->data; i++)//call for space
{
    temp = (LinkList)malloc(sizeof(LNode));
    if (temp == NULL)
    {
        printf("*Space Call Error\n");
        return ERROR;
    }
    temp->next = NULL;
    temp->data = semp_a->data;
    pos->next = temp;
    pos = pos->next;
    semp_a = semp_a->next;
}
for (int i = 0; i < Lb->data; i++)//add element
{
    temp = (LinkList)malloc(sizeof(LNode));
    if (temp == NULL)
    {
        printf("*Space Call Error\n");
        return ERROR;
    }
    temp->next = NULL;
    temp->data = semp_b->data;
```

```
    pos->next = temp;
    pos = pos->next;
    semp_b = semp_b->next;
}
printf("*ListMerge Success\n");
if (ListSort(Lc) == OK)//ListSort
    return OK;
else
    return ERROR;
}

case 13:
    printf("Your choise:13\n");
    printf("/*\n");
    printf(" *Function Name:Save\n");
    printf(" *Module:Data structures\n");
    printf(" *Use:save data in the LinearList as a file\n");
    printf("*/\n");
    //Error Alarm
    if (*L == NULL)//LinearList is null
    {
        printf("*The LinearList do not exist\n");
        getchar();
        getchar();
        break;
    }
    //save
    printf("*FileName:");
    scanf_s("%s",
writeFileName,sizeof(writeFileName));//input filename
    if ((fp = fopen(writeFileName, "wb")) == NULL)
```

```
{  
    printf("*File open error\n ");  
    getchar();  
    getchar();  
    break;  
}  
temp = *L;  
while (temp)  
{  
    fwrite(&temp->data, sizeof(ElemType), 1, fp);  
    temp = temp->next;  
}  
fclose(fp);  
printf("*Save File Success\n");  
getchar();  
getchar();  
break;  
case 14:  
    printf("Your choise:14\n");  
    printf("/*\n");  
    printf(" *Function Name:Read\n");  
    printf(" *Module:Data structures\n");  
    printf(" *Use:read data in the LinearList as a file\n");  
    printf("*\n");  
    //Error Alarm  
    if (*L == NULL)//LinearList is null  
{  
        printf("*The LinearList do not exist\n");  
        getchar();
```

```
        getchar();
        break;
    }
//read
printf("*FileName:");
scanf_s("%s",
readFileName,sizeof(readFileName));//input filename
if ((fp = fopen(readFileName, "rb")) == NULL)
{
    printf("*File open error\n ");
    getchar();
    getchar();
    break;
}
temp = *L;
fread(&temp->data, sizeof(ElemType), 1, fp);
j = temp->data;
while (j > 0)//call for space
{
    semp = (LinkList)malloc(sizeof(LNode));
    if (semp == NULL)
    {
        printf("*space call error\n");
        exit(1);
    }
    semp->next = NULL;
    temp->next = semp;
    temp = semp;
    j--;
}
```

```
temp = (*L)->next;
j = (*L)->data;
while (j > 0&&temp)//read all element
{
    fread(&temp->data, sizeof(ElemType), 1, fp);
    temp = temp->next;
    j--;
}
fclose(fp);
printf("*Read File Success\n");
getchar();
getchar();
break;
```

附录 C 基于二叉链表二叉树实现的源程序

```
BiTPos CreateBiTree1(int* pre_start, int* pre_end, int* in_start, int*  
in_end, int&size)  
{  
    int root_value = pre_start[0];  
    BiTPos root = (BiTPos)malloc(sizeof(struct BiTNode));  
    if (!root)  
    {  
        printf("*OVERFLOW\n");  
        exit(OVERFLOW);  
    }  
    root->lchild = NULL;  
    root->rchild = NULL;  
    root->index = root_value;  
    root->value = 0;  
    size++;  
  
    if (pre_start == pre_end)  
    {  
        if ((in_start == in_end) && (*pre_start == *in_end))  
        {  
            return root;  
        }  
        else  
        {  
            printf("*Input error\n");  
            return NULL;  
        }  
    }  
}
```

```
int* root_inorder = in_start;
bool is_find_root = false;
while (root_inorder <= in_end)
{
    if (*root_inorder == root_value)
    {
        is_find_root = true;
        break;
    }
    root_inorder++;
}

if (!is_find_root)
{
    printf("*Input error\n");
    return NULL;
}

int left_length = root_inorder - in_start;
int* left_preorder_end = pre_start + left_length;

if (left_length > 0)
{
    root->lchild = CreateBiTree1(pre_start + 1, left_preorder_end,
in_start, root_inorder - 1, size);
}
```

```
int* right_preorder_start = left_preorder_end + 1;
int right_length = pre_end - right_preorder_start + 1;

if (right_length > 0)
{
    root->rchild = CreateBiTree1(right_preorder_start, pre_end,
root_inorder + 1, in_end, size);
}

return root;
}

BiTPos CreateBiTree2(int* post_start, int* post_end, int* in_start, int*
in_end, int&size)
{

int root_value = *post_end;
BiTPos root = (BiTPos)malloc(sizeof(struct BiTNode));
if (!root)
{
    printf("*OVERFLOW\n");
    exit(OVERFLOW);
}
root->lchild = NULL;
root->rchild = NULL;
root->index = root_value;
root->value = 0;
size++;

if (post_start == post_end)
```

```
{  
    if ((in_start == in_end) && (*post_end == *in_end))  
    {  
        return root;  
    }  
    else  
    {  
        printf("*Input error\n");  
        return NULL;  
    }  
}
```

```
int* root_inorder = in_start;  
bool is_find_root = false;  
while (root_inorder <= in_end)  
{  
    if (*root_inorder == root_value)  
    {  
        is_find_root = true;  
        break;  
    }  
    root_inorder++;  
}  
  
if (!is_find_root)  
{  
    printf("*Input error\n");  
    return NULL;
```

```
}

int left_length = root_inorder - in_start;
int* left_postorder_end = post_start + left_length-1;

if (left_length > 0)
{

    root->lchild = CreateBiTree2(post_start, left_postorder_end,
in_start, root_inorder - 1, size);

}

int* right_postorder_start = left_postorder_end + 1;
int right_length = post_end - right_postorder_start;

if (right_length > 0)
{
    root->rchild = CreateBiTree2(right_postorder_start, post_end
- 1, root_inorder + 1, in_end, size);
}

return root;
}

void CreateBiTree3(BiTPos&root, int &size,int&ch)
{
    scanf("%d",&ch);//input an node
    printf("*input success\n");
    if (ch ==0)root = NULL;
```

```
else
{
    root = (BiTPos)malloc(sizeof(struct BiTNode));//Generate root
node
    if (!root)
    {
        printf("*OVERFLOW\n");
        exit(OVERFLOW);
    }
    size++;
    root->index = 0;
    root->lchild = NULL;
    root->rchild = NULL;
    root->value = ch;
    CreateBiTree3(root->lchild, size, ch);//Create left tree
    CreateBiTree3(root->rchild, size, ch);//Create right tree
}
}
```

```
void CreateBiTree4(BiTPos&root,int&size,ElemType
Array[],int&count)
```

```
{
    ElemtType item=Array[count-1];
    count--;
    if(item==0)
    {
        root=NULL;
    }
    else
    {
```

```
root=(BiTPos)malloc(sizeof(struct BiTNode));
if(!root)
{
    printf("*OVERFLOW\n");
    exit(OVERFLOW);
}
root->value=item;
root->index=0;
root->lchild=NULL;
root->rchild=NULL;
size++;
CreateBiTree4(root->rchild,size,Array,count);//create right
tree
CreateBiTree4(root->lchild,size,Array,count);//create light tree
}
}

//树形打印二叉树
status Show(BinaryTreePos&T)
{
if(T==NULL)
{
    printf("*The BiTree is null\n");
    return ERROR;
}
BiTPos Qe[MaxSize];
int rear=0;
int head=0;
int Depth=BiTreeDepth(T->root);
int preDepth=1;
int nowDepth=1;
```

```
int i=0;
int j=0;

int printList[MaxSize];
Qe[rear++]=T->root;
printList[i++]=T->root->index;
int posh=1;
for(j=0;j<Depth;j++)
    posh*=2;
posh--;
j=0;
while(rear!=head&&i<=posh)
{
    if(Qe[head]->lchild)
    {
        printList[i++]=Qe[head]->lchild->index;
    }
    else
    {
        Qe[head]->lchild=(BiTPos)malloc(sizeof(BiTNode));
        Qe[head]->lchild->index=-1;
        Qe[head]->lchild->lchild=NULL;
        Qe[head]->lchild->rchild=NULL;
        printList[i++]=-1;
    }
    Qe[rear++]=Qe[head]->lchild;
    if(Qe[head]->rchild)
    {
        printList[i++]=Qe[head]->rchild->index;
    }
}
```

```
    }
else
{
    Qe[head]->rchild=(BiTPos)malloc(sizeof(BiTNode));
    Qe[head]->rchild->index=-1;
    Qe[head]->rchild->lchild=NULL;
    Qe[head]->rchild->rchild=NULL;
    printList[i++]=-1;
}
Qe[rear++]=Qe[head]->rchild;
head++;
}
/*for(j=0;j<i;j++)
{
    printf("%d, ",printList[j]);
}
*/
int k=0;
for(i=1;i<=Depth;i++)
{
    //print_(10);
    if(i==1)
    {
        print_(power(2,Depth+1-i)-1);
        if(printList[k]==-1)
        {
            printf("=");
            k++;
        }
    }
    else
```

```
//printf("%d",printList[k++]);
printf("\033[30;31m%d\033[0m",printList[k++]);
print_(power(2,Depth+1-i)-1);
//print_(10);
printf("\n");
}

else
{
    print_(power(2,Depth+1-i)-1);
    for(j=0;j<power(2,i-1)-1;j++)
    {
        if(printList[k]==-1)
        {
            printf("=");
            k++;
        }
        else
            //printf("%d",printList[k++]);
            printf("\033[30;31m%d\033[0m",printList[k++]);

    print_((power(2,Depth+1)-power(2,Depth-i+2)-1)/(power(2,i-1)-1));
    }

    if(printList[k]==-1)
    {
        printf("=");
        k++;
    }
    else
        //printf("%d",printList[k++]);
```

```
    printf("\033[30;31m%d\033[0m",printList[k++]);  
    print_(power(2,Depth-i+1)-1);  
    //print_(10);  
    printf("\n");  
}  
print_(power(2,Depth+1)-1);  
printf("\n");  
  
}  
}
```

附录 D 基于邻接表图实现的源程序

```

status InsertVex(GPos&G, vexnode*v)
{
    //Graph NULL
    if (G == NULL)
    {
        printf("*The Graph struct is NULL\n");
        return ERROR;
    }
    if (G->first_vex == NULL)
    {
        printf("*The Graph is NULL\n");
        return ERROR;
    }
    vexnode*vex_pos = G->first_vex;
    while (vex_pos->nextvex&&vex_pos)
        vex_pos = vex_pos->nextvex;
    vex_pos->nextvex = v;
    printf("*Insert vex success\n");
    return OK;
}

status DeleteVex(GPos&G, int v)
{
    //Graph NULL
    if (G == NULL)
    {
        printf("*The Graph struct is NULL\n");
        return ERROR;
    }
    if (G->first_vex == NULL)
    {
        printf("*The Graph is NULL\n");
        return ERROR;
    }
    vexnode*vex_pos = G->first_vex;
    while (vex_pos&&vex_pos->index != v)
        vex_pos = vex_pos->nextvex;
    if (vex_pos == NULL)
    {
        printf("*Not Found\n");
        return ERROR;
    }
}

```

```
}

printf("*Found\n");
DeleteLink(vex_pos->first_arc);
vex_pos->first_arc = NULL;
if (vex_pos==G->first_vex)
{
    //the first vex node
    printf("*the first vex node\n");
    G->first_vex = vex_pos->nextvex;
    free(vex_pos);
    vex_pos = NULL;
}
else
{
    vexnode*vex_pre = G->first_vex;
    while (vex_pre->nextvex&&vex_pre->nextvex->index != v)
        vex_pre = vex_pre->nextvex;
    vex_pre->nextvex = vex_pos->nextvex;
    free(vex_pos);
    vex_pos = NULL;
}
vex_pos = G->first_vex;
arcnode*arc_pos = NULL;
arcnode*arc_pre = NULL;
while (vex_pos)
{
    arc_pos = vex_pos->first_arc;
    while (arc_pos)
    {
        if (arc_pos->vex_index == v)
        {
            if (arc_pos == vex_pos->first_arc)
            {
                //the first arc node
                printf("*the first arc node\n");
                vex_pos->first_arc = arc_pos->nextarc;
                free(arc_pos);
                arc_pos = vex_pos->first_arc;
            }
            else
            {
                arc_pre->nextarc = arc_pos->nextarc;
                free(arc_pos);
                arc_pos = arc_pre->nextarc;
            }
        }
    }
}
```

```

        }
        continue;
    }
    arc_pre = arc_pos;
    arc_pos = arc_pos->nextarc;
}
vex_pos = vex_pos->nextvex;

}

printf("*Delete vex success\n");
return OK;
}

status InsertArc(GPos&G, int v, int w)
{
//Graph NULL
if (G == NULL)
{
    printf("*The Graph struct is NULL\n");
    return ERROR;
}
if (G->first_vex == NULL)
{
    printf("*The Graph is NULL\n");
    return ERROR;
}
vexnode*vex_pos = G->first_vex;
while (vex_pos&&vex_pos->index != w)
    vex_pos = vex_pos->nextvex;
if (!vex_pos)
{
    printf("*the node w dont exist\n");
    return ERROR;
}
printf("*Found w\n");
vex_pos = G->first_vex;
while (vex_pos&&vex_pos->index != v)
    vex_pos = vex_pos->nextvex;
if (!vex_pos)
{
    printf("*the node v dont exist\n");
    return ERROR;
}

```

```

printf("*Found v\n");
arcnode*arc_pos = vex_pos->first_arc;
if (arc_pos == NULL)
{
    //dont have arc node
    printf("*dont have arc node yet\n");
    arc_pos = (arcnode*)malloc(sizeof(arcnode));
    if (arc_pos == NULL)
    {
        printf("*overflow\n");
        exit(OVERFLOW);
    }
    arc_pos->nextarc = NULL;
    arc_pos->vex_index = w;
    arc_pos->weight = 1;
    vex_pos->first_arc = arc_pos;
}
else
{
    while (arc_pos&&arc_pos->nextarc)
        arc_pos = arc_pos->nextarc;
    arc_pos->nextarc = (arcnode*)malloc(sizeof(arcnode));
    arc_pos->nextarc->nextarc = NULL;
    arc_pos->nextarc->vex_index = w;
    arc_pos->weight = 1;
}
printf("*Insert arc success\n");
return OK;
}

status DeleteArc(GPos&G, int v, int w)
{
//Graph NULL
if (G == NULL)
{
    printf("*The Graph struct is NULL\n");
    return ERROR;
}
if (G->first_vex == NULL)
{
    printf("*The Graph is NULL\n");
    return ERROR;
}
vexnode*vex_pos = G->first_vex;

```

```

while (vex_pos&&vex_pos->index != v)
    vex_pos = vex_pos->nextvex;
if (!vex_pos)
{
    printf("*the vex node v dont exist\n");
    return ERROR;
}
printf("*Found v\n");
if (vex_pos->first_arc == NULL)
{
    printf("*dont have v_w\n");
    return ERROR;
}
arcnode*arc_pos = vex_pos->first_arc;
while (arc_pos&&arc_pos->vex_index != w)
    arc_pos = arc_pos->nextarc;
if (!arc_pos)
{
    printf("*dont have v_w\n");
    return ERROR;
}
printf("*Found v_w\n");
if (arc_pos == vex_pos->first_arc)
{
    //the first arc node
    vex_pos->first_arc = arc_pos->nextarc;
    free(arc_pos);
    arc_pos = NULL;
    printf("*Delete arc success\n");
    return OK;
}
arcnode*arc_pre = vex_pos->first_arc;
while      (arc_pre&&arc_pre->nextarc&&arc_pre->nextarc-
>vex_index != w)
    arc_pre = arc_pre->nextarc;
arc_pre->nextarc = arc_pos->nextarc;
free(arc_pos);
arc_pos = NULL;
printf("*Delete arc success\n");
return OK;
}

void DFS(GPos&G, vexnode*vex, void(*visit)(vexnode*vex))
{

```

```

if (visited[vex->index] == 1) return;
visited[vex->index] = 1;
visit(vex);
arcnode*arc_pos = vex->first_arc;
vexnode*vex_pos = NULL;
while (arc_pos)
{
    if (visited[arc_pos->vex_index] == 0)
    {
        vex_pos = G->first_vex;
        while (vex_pos&&vex_pos->index != arc_pos->vex_index)
            vex_pos = vex_pos->nextvex;
        if (vex_pos == NULL)
        {
            printf("*the Graph is wrong\n");
            exit(-1);
        }
        DFS(G, vex_pos, visit);
    }
    arc_pos = arc_pos->nextarc;
}
}

void BFS(GPos&G, void(*visit)(vexnode*vex))
{
    int head = 0;
    int rear = 0;
    vexnode*vex_pos = G->first_vex;
    arcnode*arc_pos = NULL;
    vexnode*vex_temp = NULL;
    vexnode*vex_find = NULL;
    while (vex_pos)
    {
        if (visited[vex_pos->index] == 0)
        {
            visited[vex_pos->index] = 1;
            visit(vex_pos);
            Queue[rear++] = vex_pos;
            while (rear != head)
            {
                vex_temp = Queue[head];
                head++;
                arc_pos = vex_temp->first_arc;
                while (arc_pos)

```

```

    {
        if (visited[arc_pos->vex_index] == 0)
        {
            visited[arc_pos->vex_index] = 1;
            vex_find = G->first_vex;
            while (vex_find&&vex_find->index != arc_pos->vex_index)
                vex_find = vex_find->nextvex;
            if (vex_find == NULL)
            {
                printf("*the Graph is wrong\n");
                exit(-1);
            }
            visit(vex_find);
            Queue[rear++] = vex_find;
        }
        arc_pos = arc_pos->nextarc;
    }
}

vex_pos = vex_pos->nextvex;
}
}

```

case 13:

```

printf("Your choise:13\n");
printf("/*\n");
printf(" *Function Name:save\n");
printf(" *Module:Data structures\n");
printf(" *Use:save this Graph as a file\n");
if((*G)->first_vex==NULL)
{
    printf("*The Graph is null\n");
    getchar();
    getchar();
    break;
}
for(i=0;i<max_vex;i++)
{

```

```
vex_save[i]=-1;
}
for(i=0;i<max_arc;i++)
{
    arc_save[i]=-1;
}
printf("*file name:");
scanf("%s",writeFileName);
if((fp=fopen(writeFileName,"wb"))==NULL)
{
    printf("*File open error\n");
    getchar();
    getchar();
    break;
}
i=0;
j=0;
vex_pos=(*G)->first_vex;
arc_pos=NULL;
while(vex_pos)
{
    vex_save[i]=vex_pos->index;
    i++;
    arc_pos=vex_pos->first_arc;
    while(arc_pos)
    {
        arc_save[j]=vex_pos->index;
        arc_save[j+1]=arc_pos->vex_index;
        j+=2;
        arc_pos=arc_pos->nextarc;
    }
    vex_pos=vex_pos->nextvex;
}
i=0;
while(vex_save[i]!=-1)printf("%d\t",vex_save[i++]);
printf("\n");
j=0;
while(arc_save[j]!=-1)
{
    printf("%d,%d\t",arc_save[j],arc_save[j+1]);
    j+=2;
}
i=0;
while(vex_save[i]!=-1)
```

```
{  
    fwrite(vex_save+i,sizeof(int),1,fp);  
    i++;  
}  
fwrite(vex_save+i,sizeof(int),1,fp);  
j=0;  
while(arc_save[j]!=-1)  
{  
    fwrite(arc_save+j,sizeof(int),1,fp);  
    j++;  
}  
fwrite(arc_save+j,sizeof(int),1,fp);  
printf("*write success\n");  
fclose(fp);  
getchar();  
getchar();  
break;  
case 14:  
    printf("Your choise:14\n");  
    printf("/*\n");  
    printf(" *Function Name:Load\n");  
    printf(" *Module:Data structures\n");  
    printf(" *Use:Load a Graph from a file\n");  
    printf("*/\n");  
    printf("FileName:");  
    scanf("%s",readFileName);  
    if((fp=fopen(readFileName,"rb"))==NULL)  
    {  
        printf("File open error\n");  
        getchar();  
        getchar();  
        break;  
    }  
    for(i=0;i<max_vex;i++)  
    {  
        vex_read[i]=0;  
    }  
    for(i=0;i>max_arc;i++)  
    {  
        arc_read[i]=0;  
    }  
    i=0;  
    do  
    {
```

```

        fread(vex_read+i,sizeof(int),1,fp);
        i++;
    }while(vex_read[i-1]!=-1);
    printf("*load vex success\n");
    j=0;
    do
    {
        fread(arc_read+j,sizeof(int),1,fp);
        j++;
    }while(arc_read[j-1]!=-1);
    printf("*load arc success\n");
    i=0;
    while(vex_read[i]!=-1)printf("%d\n",vex_read[i++]);
    printf("\n");
    j=0;
    while(arc_read[j]!=-1)
    {
        printf("%d,%d\t",arc_read[j],arc_read[j+1]);
        j+=2;
    }
    printf("\n");
    if(CreateGraph(*G,vex_read,arc_read)==OK)
    {
        printf("*Read Success\n");
    }
    else
    {
        printf("*Read Error\n");
    }
    getchar();
    getchar();
    break;
}

```

自主编写的 Makefile 文件

```

#Makefile for Graph
TARGET=Graph.out
OBJECT=Graph.o
SOURCE=Graph.cpp
#make
$(TARGET):$(OBJECT)
    #gcc $^ -o $@
    g++ $^ -o $@
$(OBJECT):$(SOURCE)
    #gcc -c Graph.cpp -o Graph.o

```

```
g++ -c Graph.cpp -o Graph.o  
#clean  
.PHONY:clean  
rm -rf $(OBJECT) $(TARGET)
```