

**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201702**

**学 号： U201612922**

**姓 名： 刘子煜**

**指导教师： 许贵平**

**报告日期： 2017年 1月 12 日**

**计算机科学与技术学院**

目 录

[1 基于顺序存储结构的线性表实现 1](#_Toc458159879)

[1.1 问题描述 2](#_Toc458159880)

[1.2 系统设计 4](#_Toc458159882)

[1.3 系统实现 1](#_Toc458159883)3

[1.4 实验小结 1](#_Toc458159884)8

[2 基于链式存储结构的线性表实现 1](#_Toc458159885)9

[2.1 问题描述 1](#_Toc458159886)9

[2.2 系统设计 2](#_Toc458159887)1

[2.3 系统实现 3](#_Toc458159888)2

[2.4 实验小结 3](#_Toc458159889)9

[3 基于二叉链表的二叉树实现 4](#_Toc458159890)0

[3.1 问题描述 4](#_Toc458159891)0

[3.2 系统设计 4](#_Toc458159892)2

[3.3 系统实现 5](#_Toc458159893)8

[3.4 实验小结 7](#_Toc458159893)1

[4 基于邻接表的图实现 7](#_Toc458159895)**[错误！未定义书签。](#_Toc458159895)**

[4.1 问题描述 7](#_Toc458159896)**[错误！未定义书签。](#_Toc458159896)**

[4.2 系统设计 7](#_Toc458159897)4

[4.3 系统实现 8](#_Toc458159898)9

[4.4 实验小结 1](#_Toc458159899)00

[参考文献 1](#_Toc458159900)01

[附录A 基于顺序存储结构的线性表实现的源程序 1](#_Toc458159901)03

[附录B 基于顺序存储结构的多线性表实现的源程序 1](#_Toc458159902)14

[附录C 基于链式存储结构的线性表实现的源程序 1](#_Toc458159902)37

[附录D 基于二叉链表的二叉树实现的源程序 1](#_Toc458159903)47

[附录E 基于邻接表的图实现的源程序 16](#_Toc458159904)9

# 1 基于顺序存储结构的线性表实现

## 1.1 问题描述

创建基于顺序存储结构的线性表，以顺序表为物理结构表示数据元素之间的逻辑关系，用函数实现12个基本功能，并构造一个具有菜单的功能演示系统并将数据用文件的形式进行保存读取，且演示系统可选择实现多个线性表管理。

## 1.1.1 实验目的

通过实验达到

⑴加深对线性表的概念、基本运算的理解；

⑵熟练掌握线性表的逻辑结构与物理结构的关系；

⑶物理结构采用顺序表,熟练掌握线性表的基本运算的实现。

## 1.1.2 实验中需要实现的函数

以函数形式实现线性表的初始化表、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，具体运算功能定义如下。

⑴初始化表：函数名称是InitaList(L)；初始条件是线性表L不存在已存在；操作结果是构造一个空的线性表。

⑵销毁表：函数名称是DestroyList(L)；初始条件是线性表L已存在；操作结果是销毁线性表L。

⑶清空表：函数名称是ClearList(L)；初始条件是线性表L已存在；操作结果是将L重置为空表。

⑷判定空表：函数名称是ListEmpty(L)；初始条件是线性表L已存在；操作结果是若L为空表则返回TRUE,否则返回FALSE。

⑸求表长：函数名称是ListLength(L)；初始条件是线性表已存在；操作结果是返回L中数据元素的个数。

⑹获得元素：函数名称是GetElem(L,i,e)；初始条件是线性表已存在，1≤i≤ListLength(L)；操作结果是用e返回L中第i个数据元素的值。

⑺查找元素：函数名称是LocateElem(L,e,compare())；初始条件是线性表已存在；操作结果是返回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。

⑻获得前驱：函数名称是PriorElem(L,cur\_e,pre\_e)；初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是第一个，则用pre\_e返回它的前驱，否则操作失败，pre\_e无定义。

⑼获得后继：函数名称是NextElem(L,cur\_e,next\_e)；初始条件是线性表L已存在；操作结果是若cur\_e是L的数据元素，且不是最后一个，则用next\_e返回它的后继，否则操作失败，next\_e无定义。

⑽插入元素：函数名称是ListInsert(L,i,e)；初始条件是线性表L已存在且非空，1≤i≤ListLength(L)+1；操作结果是在L的第i个位置之前插入新的数据元素e。

⑾删除元素：函数名称是ListDelete(L,i,e)；初始条件是线性表L已存在且非空，1≤i≤ListLength(L)；操作结果：删除L的第i个数据元素，用e返回其值。

⑿遍历表：函数名称是ListTraverse(L,visit())，初始条件是线性表L已存在；操作结果是依次对L的每个数据元素调用函数visit()。

**1.1.3 需求分析**

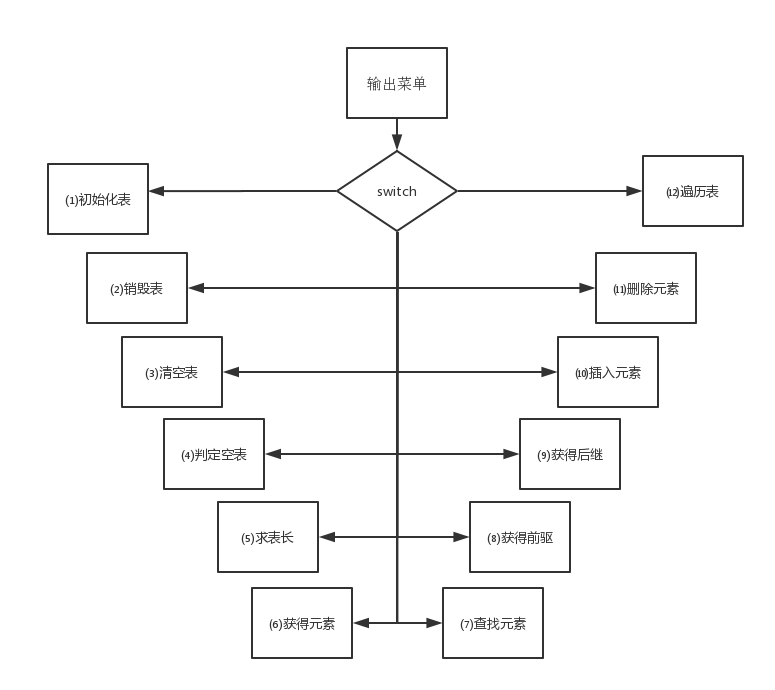
基于顺序存储结构的线性表是一种随机存取结构，存取任何元素的时间复杂度是O(1)；结构简单，逻辑上相邻的元素在物理上也相邻；不使用指针，节省存储空间；适宜于某些并行算法。

## 

## 1.2 系统设计

**1.2.1 总体设计**

实现初始化表、销毁表、清空表、判定空表、求表长和获得元素等12种基本运算并用函数进行封装，在主程序中设置菜单显示各个功能并进行标号，通过switch函数对前面运算函数进行调用以实现对应标号的功能。

图1-1模块结构图

### 1.2.2 数据结构设计

#define TRUE 1

#define FALSE 0

#define OK 1

#define SPEC 2

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

typedef int status;

typedef int ElemType;

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

typedef struct SQlist{

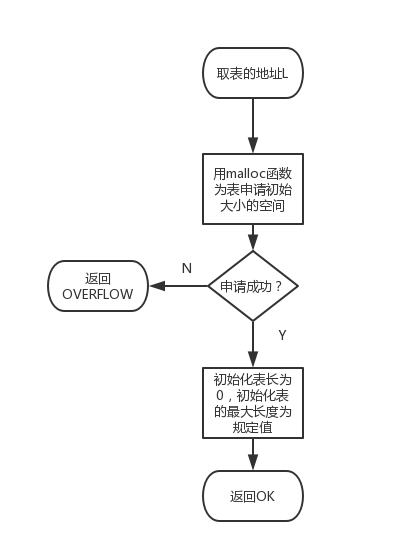
ElemType \* elem;

int length;

int listsize;

}SqList;

**1.2.3 算法设计**

**1.创建表**

**算法输入：**表指针

**算法输出：**实现状态（完成OK，失败OVERFLOW）

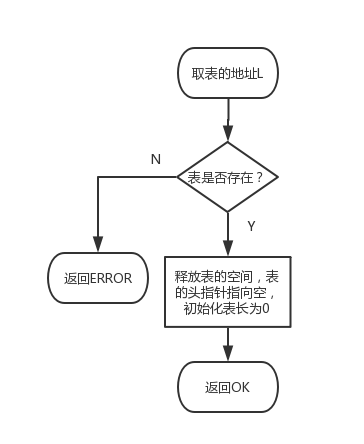
**算法处理步骤描述：**

（1）为创建表申请空间，并初始化表的基本信息。

如图1-2所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=O(1)

图1-2创建表算法流程图



**2.销毁表**

**算法输入：**表指针

**算法输出：**实现状态（完成OK，失败ERROR）

**算法处理步骤描述：**

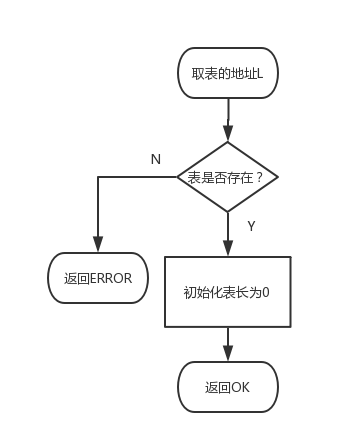
1. 如果未创建表，则返回ERROR
2. 释放表在缓冲区中的空间，表头指针赋值

为空，表长重置为空。

如图1-3所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0

图1-3销毁表算法流程图

**3.清空表**

**算法输入：**表指针

**算法输出：**实现状态（完成OK，失败ERROR）

**算法处理步骤描述：**

（1）如果未创建表，则返回ERROR

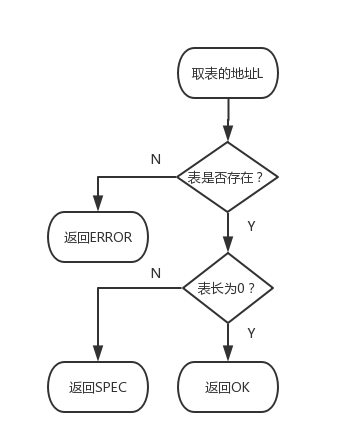
（2）表长重置为空。

如图1-4所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0

图1-4销毁表算法流程图

**4.判定空表**

**算法输入：**表

**算法输出：**实现状态（表为空OK，表不存在ERROR，表不为空SPEC）

**算法处理步骤描述：**

（1）如果未创建表，则返回ERROR

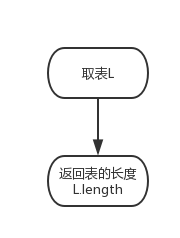
（2）表长为0则返回OK

（3）表长不为0则返回SPEC

如图1-5所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0

图1-5判定空表算法流程图

**5.求表长**

**算法输入：**表

**算法输出：**表长

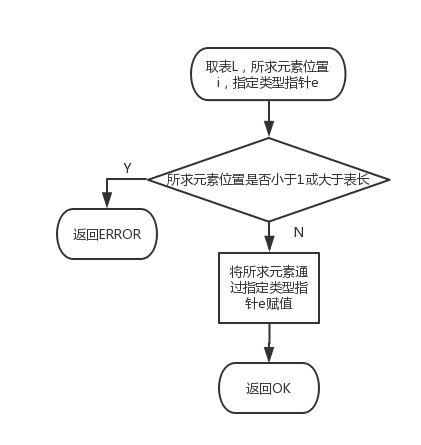
**算法处理步骤描述：**

（1）返回表长值（L.length）

如图1-6所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0

图1-6求表长算法流程图



**6.获得元素**

**算法输入：**表，所求元素位置，指定类型指针

**算法输出：**实现状态（完成OK，失败ERROR）

**算法处理步骤描述：**

1. 如果所求元素的位置小于1或超出表

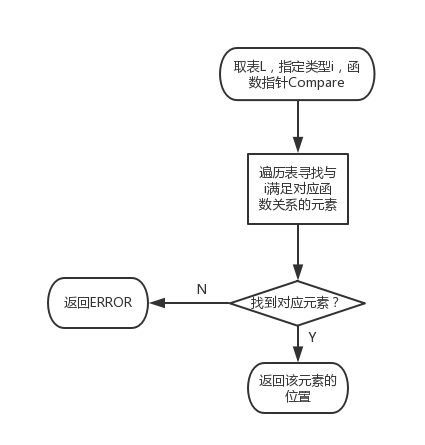
长，则返回ERROR。

（2）将所求元素通过指定类型指针进行赋值。

如图1-7所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=O(1)

图1-7获得元素算法流程图

**7.查找元素**

**算法输入：**表，整数，函数指针

**算法输出：**所定位的元素的位置（未找到返回ERROR）

**算法处理步骤描述：**

（1）遍历表中元素，寻找与整数满足函数所对应

关系的元素

（2）未找到则返回ERROR

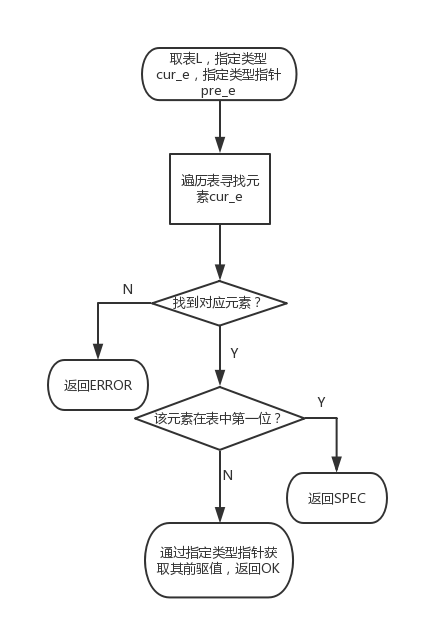
如图1-8所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=0

图1-8查找元素算法流程图

**8.获得前驱**

**算法输入：**表指针，指定类型，指定类型指针

**算法输出：**实现状态（找到OK，未找到ERROR，元素在第一位SPEC）

**算法处理步骤描述：**

1. 遍历表
2. 若找到该元素，若在第一位

则返回SPEC。

（3）若不在则通过指定类型指针

获得其前驱，返回OK

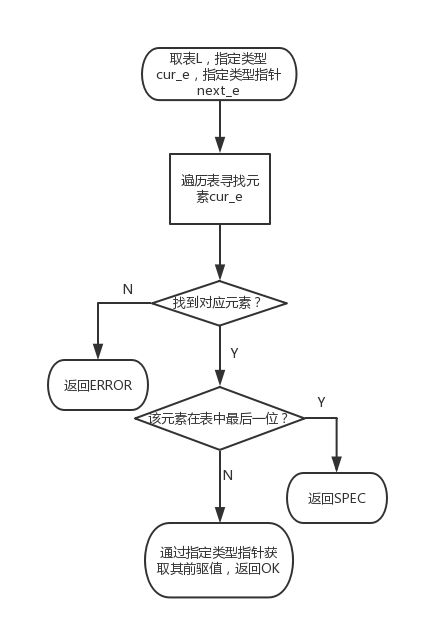
（4）若未找到则返回ERROR

如图1-9所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-9获得前驱算法流程图

**9.获得后继**

**算法输入：**表指针，指定类型，指定类型指针

**算法输出：**实现状态（找到OK，未找到ERROR，元素在最后一位SPEC）

**算法处理步骤描述：**

1. 遍历表
2. 若找到该元素，若在最后一位

则返回SPEC。

（3）若不在则通过指定类型指针

获得其前驱，返回OK

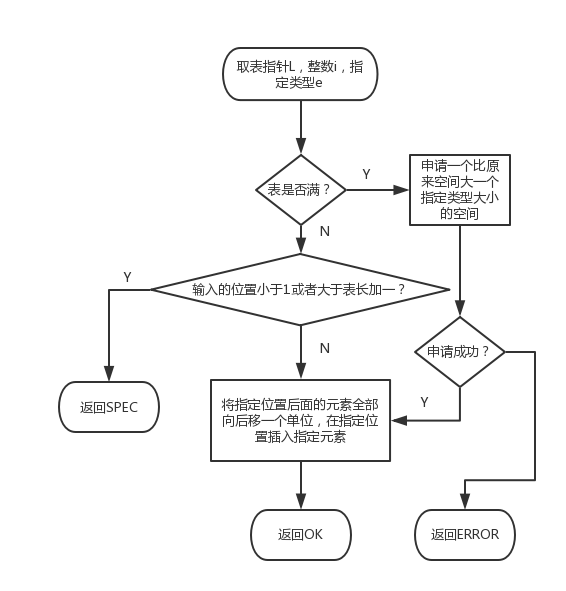
（4）若未找到则返回ERROR

如图1-10所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-10获得后继算法流程图

**10.插入元素**

**算法输入：**表指针，整数，指定类型

**算法输出：**实现状态（插入成功OK，空间申请失败ERROR，位置输入错误SPEC）

**算法处理步骤描述：**

1. 如果表已满，则申请比原来大一个指定

元素大小的表，若申请失败则返回ERROR。

1. 若成功，则将指定位置后面的元素向后

移一位，将指定元素插入指定位置表长加一，

返回OK。

1. 若表不满，若输入位置小于一或大于表

长加一，返回SPEC。

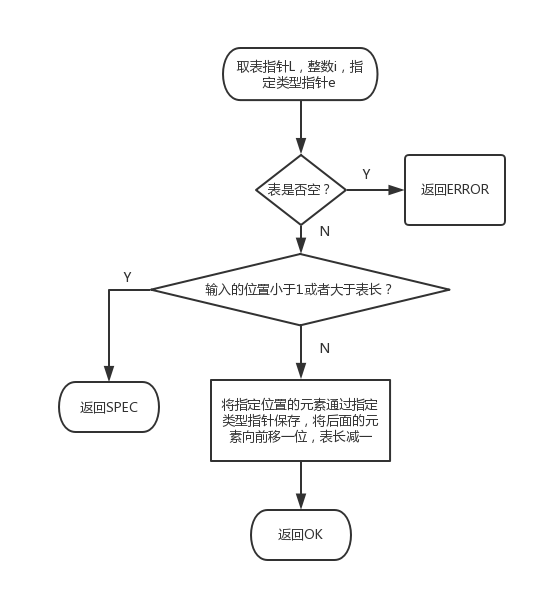
1. 若输入位置符合要求则将指定位置后面的元素向后移一位，将指定元素插入指定位置表长加一，返回OK。

如图1-11所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-11插入元素算法流程图

**11.删除元素**

**算法输入：**表指针，整数，指定类型指针

**算法输出：**实现状态（删除成功OK，表为空ERROR，位置输入错误SPEC）

**算法处理步骤描述：**

（1）如果表为空，返回ERROR。

（2）若输入位置小于一或大于表长，

返回SPEC。

1. 若输入位置符合要求则将指定

位置的元素通过指定类型指针保存，

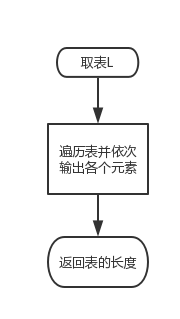
将后面的元素向前移一位，表长减一，

返回OK。

如图1-12所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-12删除元素算法流程图



**12.遍历表**

**算法输入：**表

**算法输出：**表的长度

**算法处理步骤描述：**

（1）遍历表并依次将每个元素输出

如图1-13所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-13遍历表算法流程图

## 1.3 系统实现

## 程序开发程序为Dev-c，实现语言为C语言。

主要函数为：

status IntiaList(SqList \*L);

status DestroyList(SqList \*L);

status ClearList(SqList \*L);

status ListEmpty(SqList L);

int ListLength(SqList L);

status GetElem(SqList L,int i,ElemType \*e);

int LocateElem(SqList L,ElemType e,status (\*Compare) (ElemType x,ElemType y));

status PriorElem(SqList L,ElemType cur,ElemType \*pre\_e);

status NextElem(SqList L,ElemType cur,ElemType \*next\_e);

status ListInsert(SqList \*L,int i,ElemType e);

status ListDelete(SqList \*L,int i,ElemType \*e);

status ListTrabverse(SqList L);

status Compare(ElemType x,ElemType y);

主函数中通过switch来实现输入不同的数字调用不同的功能并调用其对应的函数。

本程序中包含文件的读写功能，对应于功能的十三和十四。

多表用另一程序实现。

**测试**

测试文件为1到100的表

**示例1：**

对求表长功能的测试

测试用例及结果如表1-1所示

表1-1求表长测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 求表长失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 空表 | 返回表长为0 | IMG_256  符合理论结果 |
| 用例三 | 1到100的表 | 返回表长为100 | 9[U][YZ2%Y804`4IWU%NPQWIMG_256  符合理论结果 |

**示例2：**

对获得元素功能的测试

测试用例及结果如表1-2所示

表1-2获得元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 获得元素失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 所求元素位置小于1或大于表长 | 获取失败，给出提示 | IMG_256  符合理论结果 |
| 用例三 | 符合位置要求，50 | 获取第50位元素 | IMG_256  符合理论结果 |

**示例3：**

对查找元素功能的测试

测试用例及结果如表1-3所示

表1-3查找元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 查找元素失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 未找到对应元素，101 | 查找元素失败给出提示 | IMG_256  符合理论结果 |
| 用例三 | 存在对应元素，50（关系为大于） | 返回第一个大于50的元素的位置 | IMG_256  符合理论结果 |

**示例4：**

对获得前驱功能的测试

测试用例及结果如表1-4所示

表1-4获得前驱测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 获得前驱失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 所求元素在第一位 | 无前驱，给出提示 | IMG_256  符合理论结果 |
| 用例三 | 元素不在第一位，50 | 获取前驱成功，给出提示和前驱元素的值 | IMG_256  符合理论结果 |

**示例5：**

对获得后继功能的测试

测试用例及结果如表1-5所示

表1-5获得后继测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 获得后继失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 所求元素在最后一位 | 无后继，给出提示 | IMG_256  符合理论结果 |
| 用例三 | 元素不在最后一位，50 | 获取后成功，给出提示和后继元素的值 | IMG_256  符合理论结果 |

**示例6：**

对插入元素功能的测试

测试用例及结果如表1-6所示

表1-6插入元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 插入元素失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 输入位置小于1或大于表长加一 | 位置错误，给出提示 | IMG_256  符合理论结果 |
| 用例三 | 位置符合要求，100,101 | 将元素插入到相应位置，其后元素向后移 | IMG_256  符合理论结果 |

**示例7：**

对删除元素功能的测试

测试用例及结果如表1-7所示

表1-7删除元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 插入元素失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 输入位置小于1或大于表长 | 位置错误，给出提示 | IMG_256  符合理论结果 |
| 用例三 | 位置符合要求，50 | 将对应位置的元素删除并返回其值，其后元素向前移 | IMG_256  符合理论结果 |

**示例8：**

对遍历表功能的测试

测试用例及结果如表1-8所示

表1-8遍历表测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 插入元素失败给出提示 | IMG_256  符合理论结果 |
| 用例二 | 表为空 | 元素输出为空给出提示 | IMG_256  符合理论结果 |
| 用例三 | 表不为空 | 将表中元素依次输出 | IMG_256  符合理论结果 |

经过实验，程序整体测试效果良好，程序基本满足实验要求。

## 1.4 实验小结

# 本次实验整体较为简单基础，在实验过程中并没有遇到太大的问题，不过开始时在一些细节的条件判断上出现了些许问题，导致程序出现了一些BUG，经过简单的调试现已修复，关于多表的操作，有了已经完成的封装好的函数实现多表并不困难，在每个功能之前添加一些判断并选择想要操作的表的名称基本就能实现，不过在多表的文件读写上出现了一些问题，花费了一些时间才得以解绝。总体来说，本次实验是数据结构第一次实验，虽然是第一次构建表，但整体来说相对不难，不过在编程过程中出现了很多小的问题，这是在以后的编程 过程中需要注意和改正的。

# 2 基于链式存储结构的线性表实现

## 2.1 问题描述

创建基于链式存储结构的线性表，以链表为物理结构表示数据元素之间的逻辑关系，用函数实现12个基本功能，并构造一个具有菜单的功能演示系统并将数据用文件的形式进行保存读取，且演示系统可选择实现多个线性表管理。

## 2.1.1 实验目的

通过实验达到

⑴加深对线性链表的概念、基本运算的理解；

⑵熟练掌握线性链表的逻辑结构与物理结构的关系；

⑶物理结构采用链表,熟练掌握线性链表的基本运算的实现。

## 2.1.2 实验中需完成的函数

以函数形式实现线性链表的初始化表、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，具体运算功能定义如下。

⑴初始化表：函数名称是InitaList(\*\*head)；初始条件是线性链表L不存在已存在；操作结果是构造一个空的线性表。

⑵销毁表：函数名称是DestroyList(\*\*head)；初始条件是线性链表L已存在；操作结果是销毁线性表L。

⑶清空表：函数名称是ClearList(\*head)；初始条件是线性链表L已存在；操作结果是将L重置为空表。

⑷判定空表：函数名称是ListEmpty(\*head)；初始条件是线性链表L已存在；操作结果是若L为空表则返回TRUE,否则返回FALSE。

⑸求表长：函数名称是ListLength(\*head)；初始条件是线性链表已存在；操作结果是返回L中数据元素的个数。

⑹获得元素：函数名称是GetElem(\*head,i,\*e)；初始条件是线性链表已存在，1≤i≤ListLength(L)；操作结果是用e返回L中第i个数据元素的值。

⑺查找元素：函数名称是LocateElem(\*head,e,\*compare())；初始条件是线性链表已存在；操作结果是返回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。

⑻获得前驱：函数名称是PriorElem(\*head,cur\_e,\*pre\_e)；初始条件是线性链表L已存在；操作结果是若cur\_e是L的数据元素，且不是第一个，则用pre\_e返回它的前驱，否则操作失败，pre\_e无定义。

⑼获得后继：函数名称是NextElem(\*head,cur\_e,\*next\_e)；初始条件是线性链表L已存在；操作结果是若cur\_e是L的数据元素，且不是最后一个，则用next\_e返回它的后继，否则操作失败，next\_e无定义。

⑽插入元素：函数名称是ListInsert(\*head,i,e)；初始条件是线性链表L已存在且非空，1≤i≤ListLength(L)+1；操作结果是在L的第i个位置之前插入新的数据元素e。

⑾删除元素：函数名称是ListDelete(\*head,i,\*e)；初始条件是线性链表L已存在且非空，1≤i≤ListLength(L)；操作结果：删除L的第i个数据元素，用e返回其值。

⑿遍历表：函数名称是ListTraverse(\*head)，初始条件是线性链表L已存在；操作结果是依次将L中的每个数据元素输出。

**2.1.3 需求分析**

比[顺序存储结构](https://baike.baidu.com/item/%E9%A1%BA%E5%BA%8F%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84" \t "https://baike.baidu.com/item/%E9%93%BE%E5%BC%8F%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84/_blank)的[存储密度](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E5%AF%86%E5%BA%A6/3236466" \t "https://baike.baidu.com/item/%E9%93%BE%E5%BC%8F%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84/_blank)小(链式存储结构中每个结点都由数据域与指针域两部分组成，相比顺序存储结构增加了存储空间)，逻辑上相邻的节点物理上不必相邻，插入、删除灵活 (不必移动节点，只要改变节点中的指针)。

一般在计算机的硬盘中，文件都是链式存储的，多个[扇区](https://baike.baidu.com/item/%E6%89%87%E5%8C%BA/3642285" \t "https://baike.baidu.com/item/%E9%93%BE%E5%BC%8F%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84/_blank)组成一个[簇](https://baike.baidu.com/item/%E7%B0%87/13014767" \t "https://baike.baidu.com/item/%E9%93%BE%E5%BC%8F%E5%AD%98%E5%82%A8%E7%BB%93%E6%9E%84/_blank)，簇是计算机存储数据的基本单位，而一个文件是存储在多个在空间上也许并不相连的簇中的。

## 2.2 系统设计

**2.2.1 总体设计**

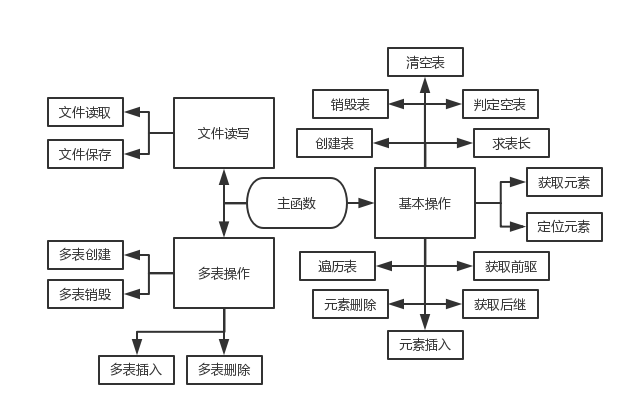
实现初始化表、销毁表、清空表、判定空表、求表长和获得元素等12种基本运算并用函数进行封装，在主程序中设置菜单显示各个功能并进行标号，通过switch函数对前面运算函数进行调用以实现对应标号的功能。同时本程序中添加了文件存储和读取作为标号13,14对应的功能，也填加了多表的创建，销毁，插入，删除和遍历作为15到19号功能，当多表已建立 ，调用前面函数对多表中某个表进行操作时，需输入其具体的名称，以确定要进行操作的表。

图1-1模块结构图

### 2.2.2 数据结构设计

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define SPEC 2

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define NAME\_MAX\_SIZE 20

#define LENG sizeof(struct Lnode)

#define SLENG sizeof(struct Slist)

typedef struct Lnode{

ElemType data;

struct Lnode \*next;

}Lnode, \*Linklist;

typedef struct Lname{

char name[NAME\_MAX\_SIZE];

Lnode \*lists;

}Lname;

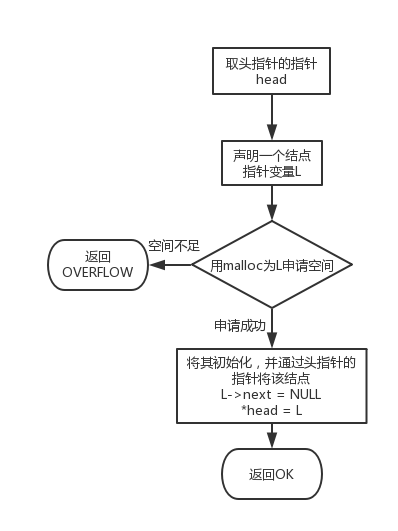
typedef struct Slist{

Lname listname;

struct Slist \*next;

}Slist, \*LinkSlist;

**2.2.3 算法设计**

**1.创建表**

**算法输入：**头指针的指针

**算法输出：**实现状态（完成OK，失败OVERFLOW）

**算法处理步骤描述：**

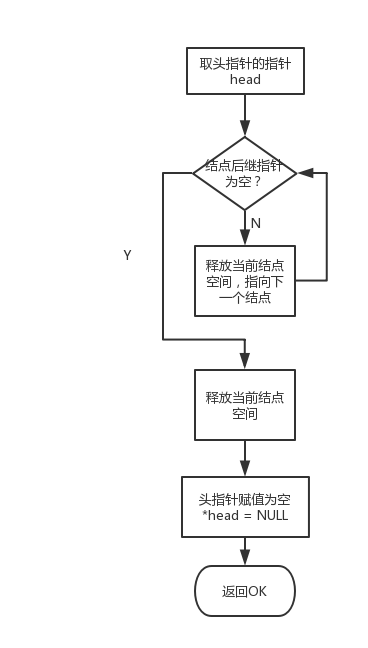
1. 声明一个结点指针变量，用malloc为其申请空间。
2. 若空间不足则返回OVERFLOW
3. 将其初始化，并通过头指针的指针将该结点

指针赋值给头指针

如图1-2所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=O(1)

图1-2创建表算法流程图



**2.销毁表**

**算法输入：**头指针的指针

**算法输出：**实现状态（完成OK）

**算法处理步骤描述：**

1. 释放表头指针及表中每个结点指针在缓冲区中的空间。
2. 头指针赋值为空

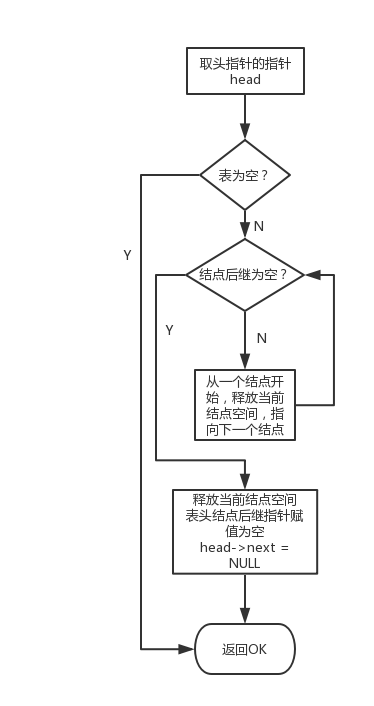
如图1-3所示。

**算法时空复杂度分析：**

T(n)=O(n);S(n)=O(1)

图1-3销毁表算法流程图

**3.清空表**



**算法输入：**头指针

**算法输出：**实现状态（完成OK）

**算法处理步骤描述：**

（1）如果表为空，返回OK

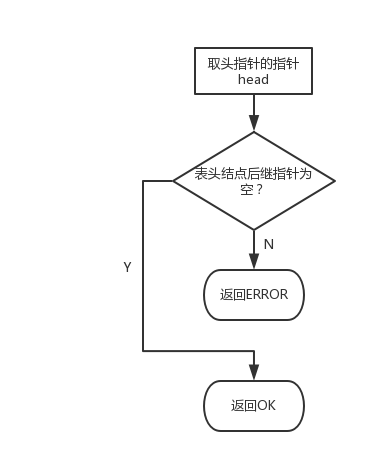
（2）释放表中每一个结点的空间

（3）将表头结点的后继指针赋值为空，返回OK

如图1-4所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-4销毁表算法流程图



**4.判定空表**

**算法输入：**头指针

**算法输出：**实现状态（表为空OK，表不为空ERROR）

**算法处理步骤描述：**

（1）表头结点后继不为空，返回EORROR

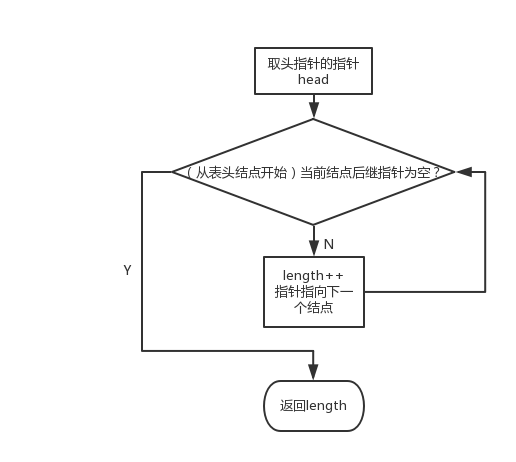
（2）表头结点后继为空，返回OK

如图1-5所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0

图1-5判定空表算法流程图

**5.求表长**



**算法输入：**头指针

**算法输出：**链表的结点个数（表长）

**算法处理步骤描述：**

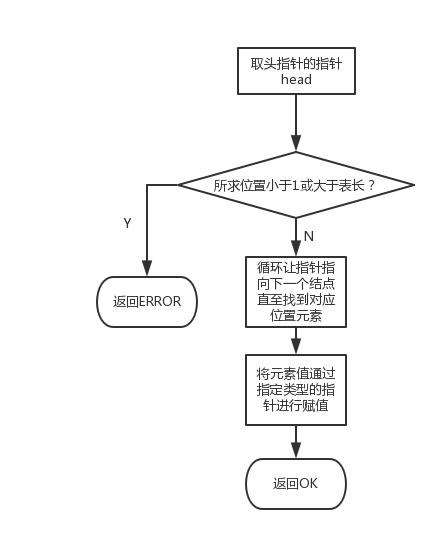
（1）遍历表，每读到一个结点表长加一

（2）返回表长值

如图1-6所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-6求表长算法流程图



**6.获得元素**

**算法输入：**头指针，所求元素位置，指定类型指针

**算法输出：**实现状态（完成OK，失败ERROR）

**算法处理步骤描述：**

1. 如果所求元素的位置小于1或超出表

长，则返回ERROR。

1. 找到链表中对应位置上的元素。
2. 将所求元素通过指定类型指针进行赋值。

如图1-7所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-7获得元素算法流程图

**7.查找元素**

**算法输入：**头指针，整数，函数指针

**算法输出：**所定位的元素的位置（未找到返回ERROR）

**算法处理步骤描述：**

1. 遍历表中元素，寻找与整数满足函数所对应关系的元素。
2. 找到则返回位置。

（3）未找到则返回ERROR。

如图1-8所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

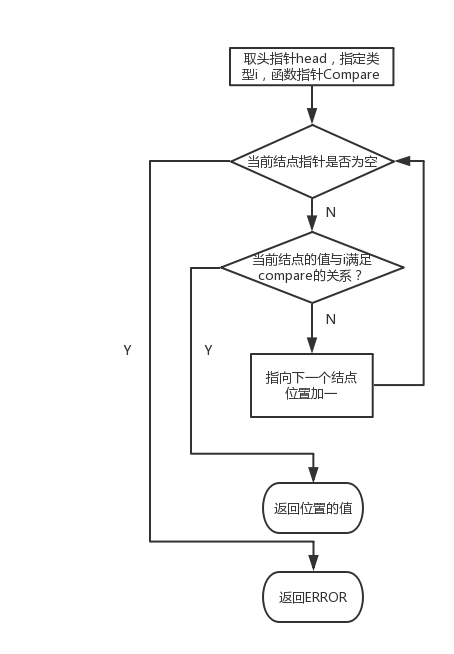


图1-8查找元素算法流程图

**8.获得前驱**

**算法输入：**头指针，指定类型，指定类型指针

**算法输出：**实现状态（找到OK，未找到ERROR，元素在第一位SPEC）

**算法处理步骤描述：**

1. 若表为空，则返回ERROR
2. 若该元素在第一位则返回SPEC。

（3）遍历链表，如果找到，则通过指定类型指针返回其前驱的元素值，返回OK

（4）若未找到则返回ERROR

如图1-9所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

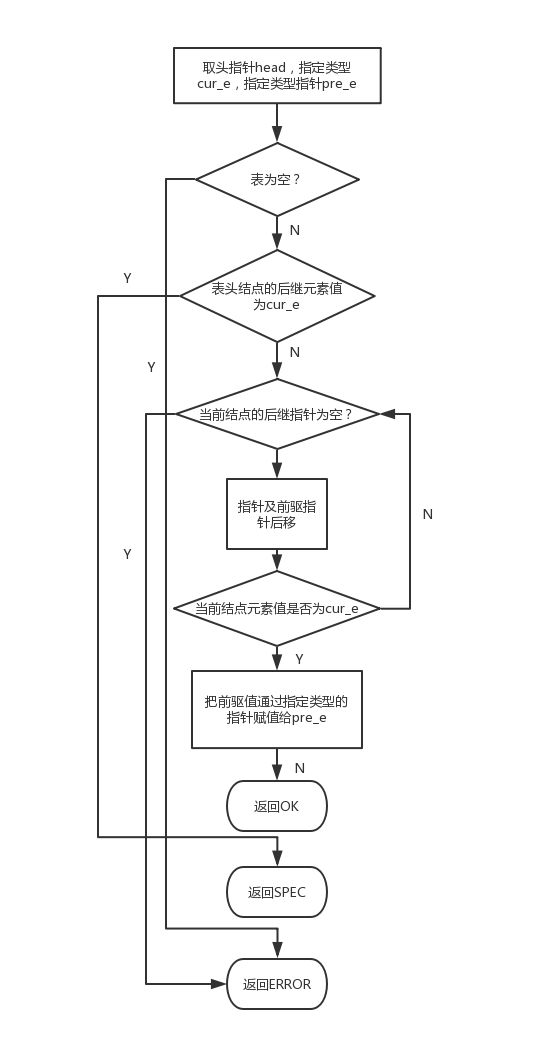


图1-9获得前驱算法流程图

**9.获得后继**

**算法输入：**头指针，指定类型，指定类型指针

**算法输出：**实现状态（找到OK，未找到ERROR，元素在最后一位SPEC）

**算法处理步骤描述：**

（1）若表为空，则返回ERROR

（2）遍历表，如果找到，则通过指定类型指针返回其前驱的元素值，返回OK

（3）若未找到，判断最后一个结点元素值是否满足要求，若满足返回SPEC

（4）若不满足则返回ERROR

如图1-10所示。

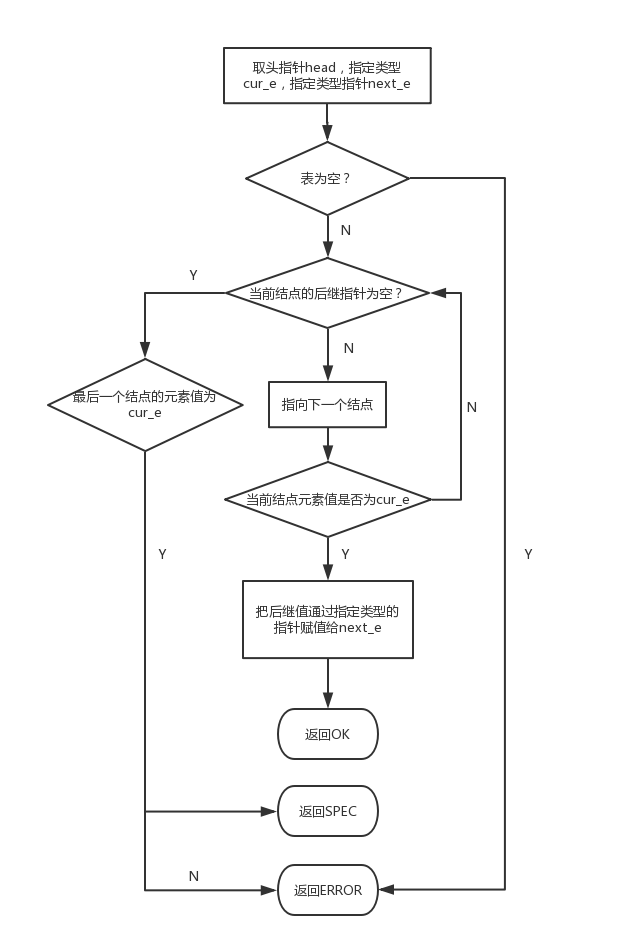
**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

图1-10获得后继算法流程图

**10.插入元素**

**算法输入：**头指针，整数，指定类型

**算法输出：**实现状态（插入成功OK，空间申请失败OVERFLOW，位置输入错误ERROR）

**算法处理步骤描述：**

1. 如果插入位置小于1或超过表长，则返回ERROR。
2. 为新建结点申请空间，若空间溢出，申请失败，则返回OVERFLOW。
3. 若申请成功，通过改变结点next值将新建结点插入到相应位置，并将元素值赋值给新插入的结点，返回OK。

如图1-11所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

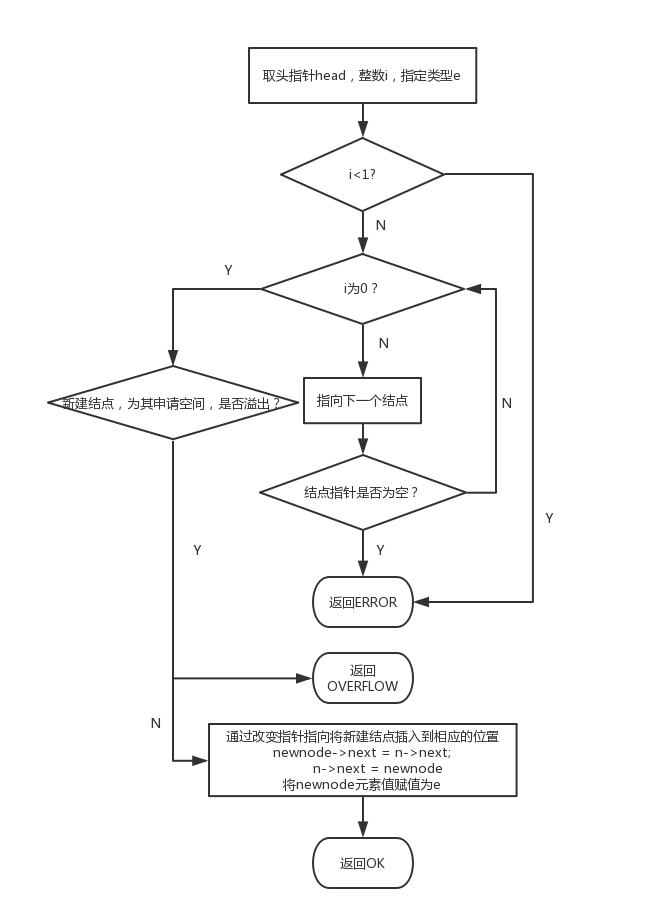


图1-11插入元素算法流程图

**11.删除元素**

**算法输入：**头指针，整数，指定类型指针

**算法输出：**实现状态（删除成功OK，位置输入错误ERROR）

**算法处理步骤描述：**

（1）若输入位置小于一或大于表长，返回ERROR。

（2）找到对应位置的结点，将其前驱的后继指针指向对应元素的后继。

（3）通过指定类型指针将结点的元素值赋给指定类型元素，释放该结点空间，返回OK。

如图1-12所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

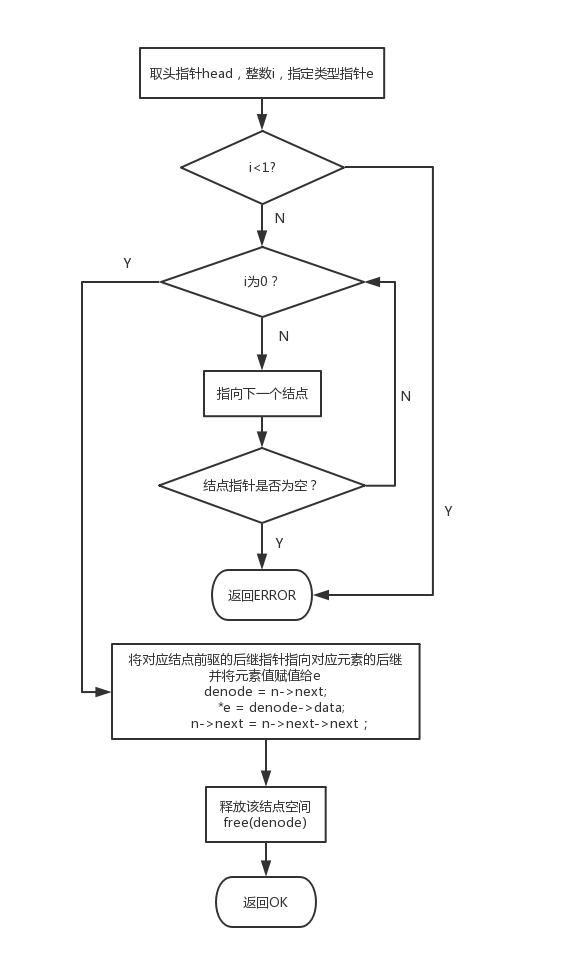


图1-12删除元素算法流程图

**12.遍历表**

**算法输入：**头结点

**算法输出：**实现状态（表为空ERROR，表不为空length）

**算法处理步骤描述：**

1. 如果表为空，则返回ERROR
2. 如果表不为空，将表中每个元素用printf函数输出，返回length

如图1-13所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)

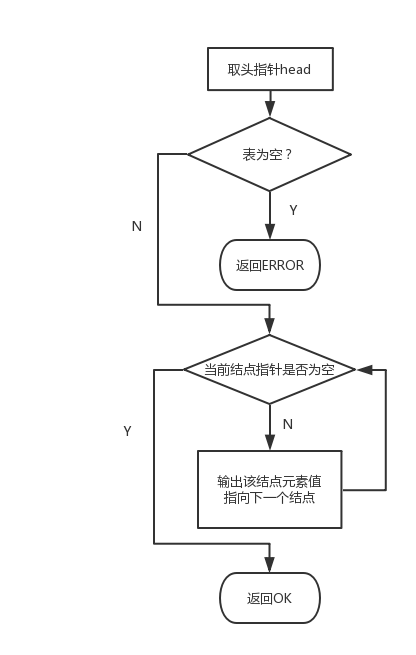


图1-13遍历表算法流程图

## 2.3 系统实现

## 程序开发程序为Dev-c，实现语言为C语言。

主要函数为：

status IntiaList(Lnode \*\*head);

status DestroyList(Lnode \*\*head);

status ClearList(Lnode \*head);

status ListEmpty(Lnode \*head);

int ListLength(Lnode \*head);

status GetElem(Lnode \*head,int i,ElemType \*e);

int LocateElem(Lnode \*head,ElemType e,status (\*Compare) (ElemType x,ElemType y));

status PriorElem(Lnode \*head,ElemType cur\_e,ElemType \*pre\_e);

status NextElem(Lnode \*head,ElemType cur\_e,ElemType \*next\_e);

status ListInsert(Lnode \*head,int i,ElemType e);

status ListDelete(Lnode \*head,int i,ElemType \*e);

status ListTrabverse(Lnode \*head);

status Compare(ElemType x,ElemType y);

status IntiaSList(Slist \*\*shead);

status DestroySList(Slist \*\*shead);

status SListInsert(Slist \*shead, int i, char name[NAME\_MAX\_SIZE]);

status SListDelete(Slist \*shead, int i, char name[NAME\_MAX\_SIZE]); 主函数中通过switch来实现输入不同的数字调用不同的功能并调用其对应的函数，基础功能对应一到十二。

本程序中包含文件的读写功能，对应于功能的十三和十四。

本程序包含多表操作功能，相应的函数有IntiaSList，DestroySList，SListInsert和SListDelete，对应于功能的十五到十八。

**测试**

测试文件为1到100的表

**示例1：**

对创建表功能的测试

测试用例及结果如表2-1所示

表2-1创建表测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 空间充足 | 创建成功  给出提示 | ([[XA5)TE_}HZ(]FG[[[C_4  符合理论结果 |
| 用例二 | 空间不足 | 创建失败  空间溢出  给出提示 |  |

**示例2：**

对销毁功能的测试

测试用例及结果如表2-2所示

表2-2销毁表测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 销毁表失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 已创建表 | 销毁成功给出提示 | E255GG9[]R8NI(BN9%NY)(9  符合理论结果 |

**示例3：**

对清空表功能的测试

测试用例及结果如表2-3所示

表2-3清空表测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 清除失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 空表 | 提示清除成功 | **WH$3K@R}VPB7(2507$2D]A9**  符合理论结果 |
| 用例三 | 1到100的表 | 提示清除成功 | 9[U][YZ2%Y804`4IWU%NPQW  符合理论结果 |

**示例4：**

对判断空表功能的测试

测试用例及结果如表2-4所示

表2-4判断空表测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 判断空表失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 空表 | 提示表为空 | 3](C@{U[EW}1908TNV)K25V  符合理论结果 |
| 用例三 | 1到100的表 | 提示表不为空 | 0@2MK}EWT{7XVH[%3[KGKT6  符合理论结果 |

**示例5：**

对求表长功能的测试

测试用例及结果如表2-5所示

表2-5求表长测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 求表长失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 空表 | 返回表长为0 | S](OLG@X[JBLAUO%U5QCXJB  符合理论结果 |
| 用例三 | 1到100的表 | 返回表长为100 | 9[U][YZ2%Y804`4IWU%NPQW34OKRH}W{{YRU~E0NWZ}%0X  符合理论结果 |

**示例6：**

对获得元素功能的测试

测试用例及结果如表2-6所示

表2-6获得元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 获得元素失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 所求元素位置小于1或大于表长 | 获取失败，给出提示 | W32Q}1OA)R3]ZU2V2UTCT8O  符合理论结果 |
| 用例三 | 符合位置要求，50 | 获取第50位元素 | W72IU{H2%7ZF$[)@AG@J`@5  符合理论结果 |

**示例7：**

对查找元素功能的测试

测试用例及结果如表2-7所示

表2-7查找元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 查找元素失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 未找到对应元素，101 | 查找元素失败给出提示 | 2)VTM]%A7PLY7%HOV3S0[7G2)VTM]%A7PLY7%HOV3S0[7G  符合理论结果 |
| 用例三 | 存在对应元素，50（关系为大于） | 返回第一个大于50的元素的位置 | QQE)B)92PF)10JIAXBU)LIPQQE)B)92PF)10JIAXBU)LIP  符合理论结果 |

**示例8：**

对获得前驱功能的测试

测试用例及结果如表2-8所示

表2-8获得前驱测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 获得前驱失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 所求元素在第一位 | 无前驱，给出提示 | [WL4O@%8GWFL}8~TKJFCS)O[WL4O@%8GWFL}8~TKJFCS)O  符合理论结果 |
| 用例三 | 元素不在第一位，50 | 获取前驱成功，给出提示和前驱元素的值 | 0LOQV[V3DVDBK{YO$P%G_)L  符合理论结果 |

**示例9：**

对获得后继功能的测试

测试用例及结果如表2-9所示

表2-9获得后继测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 获得后继失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 所求元素在最后一位 | 无后继，给出提示 | JQ_ZLZWJN8F07_EU]((I2BY  符合理论结果 |
| 用例三 | 元素不在最后一位，50 | 获取后成功，给出提示和后继元素的值 | 3XCF80QJF@A2)]A}G81V~KQ  符合理论结果 |

**示例10：**

对插入元素功能的测试

测试用例及结果如表2-10所示

表2-10插入元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 插入元素失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 输入位置小于1或大于表长加一 | 位置错误，给出提示 | L[[(4CH8NHTEE7HSO}]O)13  符合理论结果 |
| 用例三 | 位置符合要求，100,101 | 将元素插入到相应位置，其后元素向后移 | 9]}V1YCUZ6BL7HZIE(@NF@99]}V1YCUZ6BL7HZIE(@NF@9  符合理论结果 |

**示例11：**

对删除元素功能的测试

测试用例及结果如表2-11所示

表2-11删除元素测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 插入元素失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 输入位置小于1或大于表长 | 位置错误，给出提示 | K5UW~[TO$)9_YFOF6(3T~$F  符合理论结果 |
| 用例三 | 位置符合要求，50 | 将对应位置的元素删除并返回其值，其后元素向前移 | D$P%8AD4759I7~CSZ~BZ6OK  符合理论结果 |

**示例12：**

对遍历表功能的测试

测试用例及结果如表2-12所示

表2-12遍历表测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建表 | 插入元素失败给出提示 | 6D3W9[)Q{VDT51BH6~WO]@H  符合理论结果 |
| 用例二 | 表为空 | 元素输出为空给出提示 | RFZ4N%M~W~`01JJL3V[QEEP  符合理论结果 |
| 用例三 | 表不为空 | 将表中元素依次输出 | (SUQ0}(O%125XRK6_5E[RXS  符合理论结果 |

经过实验，程序整体测试效果良好，程序基本满足实验要求。

## 2.4 实验小结

本次实验由于之前多次构建链表，且实验内容操作与上一次的实验的要求差别不大，所以本次实验做起来较为轻车熟路，但是在实验过程中仍然在一些函数的构建中因为考虑不足，导致在测试过程中出现了一些问题，不过经过重新思考和修改也使得问题得以解决，在多表的构建方面，有了上次实验多表构建的经验，本次多表的构建放在了同一个程序中，与基础程序分开，当多表构建之后，对基础功能进行操作时才会体现，但是功能不算特别完善，还有不少需要改进提高的地方，在课下和以后的编程过程中，我会继续不断地改善对此类结构构建的方法。

# 

# 3基于二叉链表的二叉树实现

## 3.1 问题描述

采用二叉链表表作为二叉树的物理结构，用函数实现20个基本功能，并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。演示系统可选择实现二叉树的文件形式保存和多个二叉树管理。

## 3.1.1 实验目的

通过实验达到

⑴加深对二叉树的概念、基本运算的理解；

⑵熟练掌握二叉树的逻辑结构与物理结构的关系；

⑶以二叉链表作为物理结构，熟练掌握二叉树基本运算的实现。

## 3.1.2 实验中需完成的函数

依据最小完备性和常用性相结合的原则，以函数形式定义了二叉树的初始化二叉树、销毁二叉树、创建二叉树、清空二叉树、判定空二叉树和求二叉树深度等20种基本运算，具体运算功能定义如下。

⑴初始化二叉树：函数名称是InitBiTree(T)；初始条件是二叉树T不存在；操作结果是构造空二叉树T。

⑵销毁二叉树：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。

⑶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；操作结果是按definition构造二叉树T。

⑷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。

⑸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。

⑹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。

⑺获得根结点：函数名称是Root(T)；初始条件是二叉树T已存在；操作结果是返回T的根。

⑻获得结点：函数名称是Value(T,e)；初始条件是二叉树T已存在，e是T中的某个结点；操作结果是返回e的值。

⑼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T中的某个结点；操作结果是结点e赋值为value。

⑽获得双亲结点：函数名称是Parent(T,e) ；初始条件是二叉树T已存在，e是T中的某个结点；操作结果是若e是T的非根结点，则返回它的双亲结点指针，否则返回NULL。

⑾获得左孩子结点：函数名称是LeftChild(T,e)；初始条件是二叉树T存在，e是T中某个节点；操作结果是返回e的左孩子结点指针。若e无左孩子，则返回NULL。

⑿获得右孩子结点：函数名称是RightChild(T,e)；初始条件是二叉树T已存在，e是T中某个结点；操作结果是返回e的右孩子结点指针。若e无右孩子，则返回NULL。

⒀获得左兄弟结点：函数名称是LeftSibling(T,e)；初始条件是二叉树T存在，e是T中某个结点；操作结果是返回e的左兄弟结点指针。若e是T的左孩子或者无左兄弟，则返回NULL。

⒁获得右兄弟结点：函数名称是RightSibling(T,e)；初始条件是二叉树T已存在，e是T中某个结点；操作结果是返回e的右兄弟结点指针。若e是T的右孩子或者无有兄弟，则返回NULL。

⒂插入子树：函数名称是InsertChild(T,p,LR,c)；初始条件是二叉树T存在，p指向T中的某个结点，LR为0或1，,非空二叉树c与T不相交且右子树为空；操作结果是根据LR为0或者1，插入c为T中p所指结点的左或右子树，p 所指结点的原有左子树或右子树则为c的右子树。

⒃删除子树：函数名称是DeleteChild(T.p.LR)；初始条件是二叉树T存在，p指向T中的某个结点，LR为0或1。 操作结果是根据LR为0或者1，删除c为T中p所指结点的左或右子树。

⒄前序遍历：函数名称是PreOrderTraverse(T,Visit())；初始条件是二叉树T存在，Visit是对结点操作的应用函数；操作结果：先序遍历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。

⒅中序遍历：函数名称是InOrderTraverse(T,Visit))；初始条件是二叉树T存在，Visit是对结点操作的应用函数；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。

⒆后序遍历：函数名称是PostOrderTraverse(T,Visit))；初始条件是二叉树T存在，Visit是对结点操作的应用函数；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。

⒇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。

## 3.1.3 需求分析

二叉树通常作为数据结构应用，典型用法是对节点定义一个标记函数，将一些值与每个节点相关系，这样标记的二叉树就可以实现二叉查找树和二元堆积，并应用于高效率的搜索和排序。同时，以二叉树为基础的平衡二叉树，红黑树，哈夫曼树等结构在很多地方被广泛应用。

## 3.2 系统设计

**3.2.1 总体设计**

实现二叉树的初始化、销毁二叉树、创建二叉树、清空二叉树、判定空二叉树和求二叉树深度等20种基本运算并用函数进行封装，在主程序中设置菜单显示各个功能并进行标号，通过switch函数对前面运算函数进行调用以实现对应标号的功能。同时本程序中添加了文件存储和读取作为标号21，22对应的功能，也填加了二叉树的插入，二叉树的删除，创建新的二叉树，选择二叉树等功能实现多个二叉树的管理，作为23到26号功能。

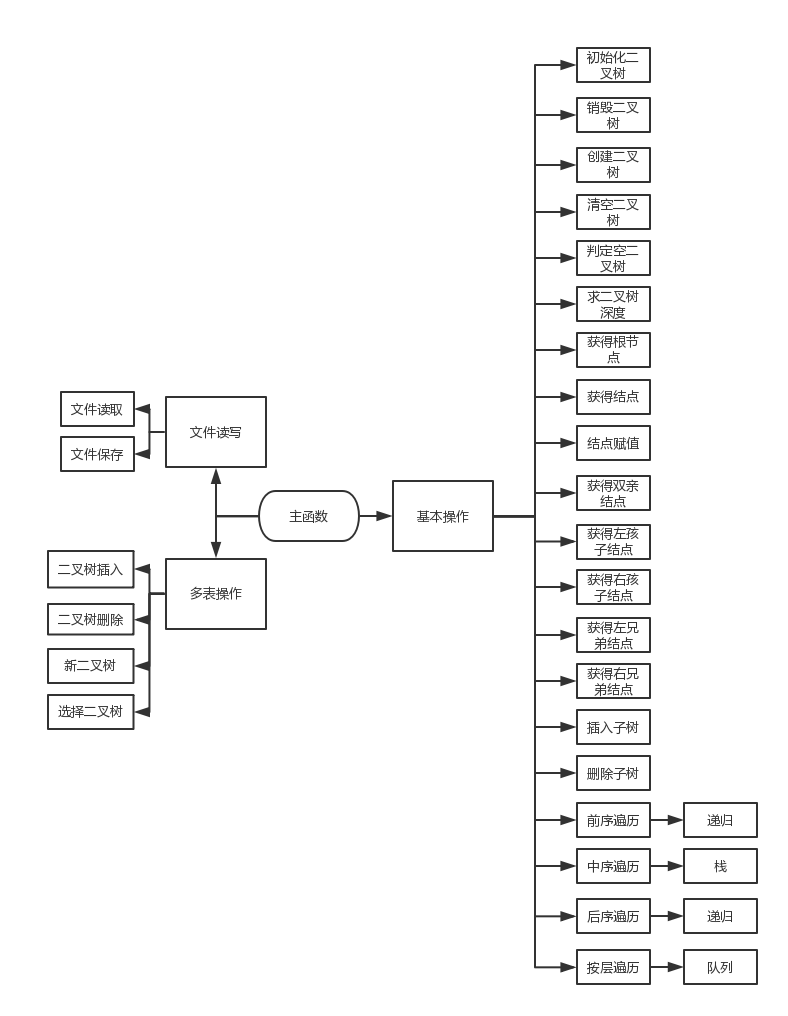


图3-1模块结构图



图3-2 多二叉树管理的物理结构示意图

### 3.2.2 数据结构设计

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define SPEC -2

#define NAME\_MAX\_SIZE 20

#define TEL\_MAX\_SIZE 15

#define TREE\_MAX\_SIZE 200

#define TREE\_NAME\_SIZE 20

#define MENU\_INIT\_SIZE 50

typedef int status;

typedef char ElemType1;

typedef char ElemType2;

typedef struct BiTNode{

ElemType1 name[NAME\_MAX\_SIZE];

ElemType2 tel[TEL\_MAX\_SIZE];

struct BiTNode \*lchild;

struct BiTNode \*rchild;

}BiTNode;

typedef struct MenuName

{

char name[TREE\_NAME\_SIZE];

BiTNode \*bitree;

}MenuName;

typedef struct Menu{

MenuName \*menuname;

int length;

int menusize;

}Menu;

int flag = 0;

static int count = 0;

#define LENG sizeof(BiTNode)

**3.2.3 算法设计**

**1.初始化二叉树**

**算法输入：**根结点指针的指针

**算法输出：**实现状态（完成OK）

**算法处理步骤描述：**

（1）将根结点指针赋值为空

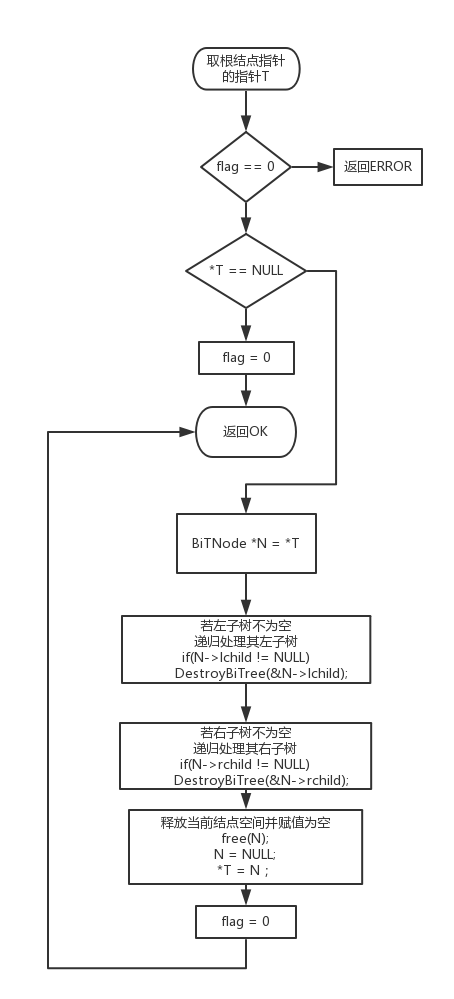
（2）flag变为1

（3）返回OK

如图3-3所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

图3-3初始化二叉树算法流程图



**2.销毁二叉树**

**算法输入：**根结点指针的指针

**算法输出：**实现状态（完成OK，未创建返回ERROR）

**算法处理步骤描述：**

（1）如果flag为0则二叉树未创建返回ERROR

（2）如果根结点指针为空，则为空树，flag赋

值为0返回OK

1. 如果左（右）子树不为空，则递归其左（右）

子树

1. 释放当前结点，赋值为空，flag赋值为0，

返回OK

如图3-4所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

图3-4销毁二叉树算法流程图

**3.创建二叉树**

**算法输入：**根结点指针的指针，标识序列

**算法输出：**实现状态（完成OK，空间不足OVERFLOW）

**算法处理步骤描述：**

（1）如果读到的标识符为‘#’则将该结点置为空

（2）为新增结点申请空间，将其标识符和元素值放入

（3）递归该函数处理新增结点的左（右）子树

（4）flag赋值为1，返回OK

如图3-4所示。

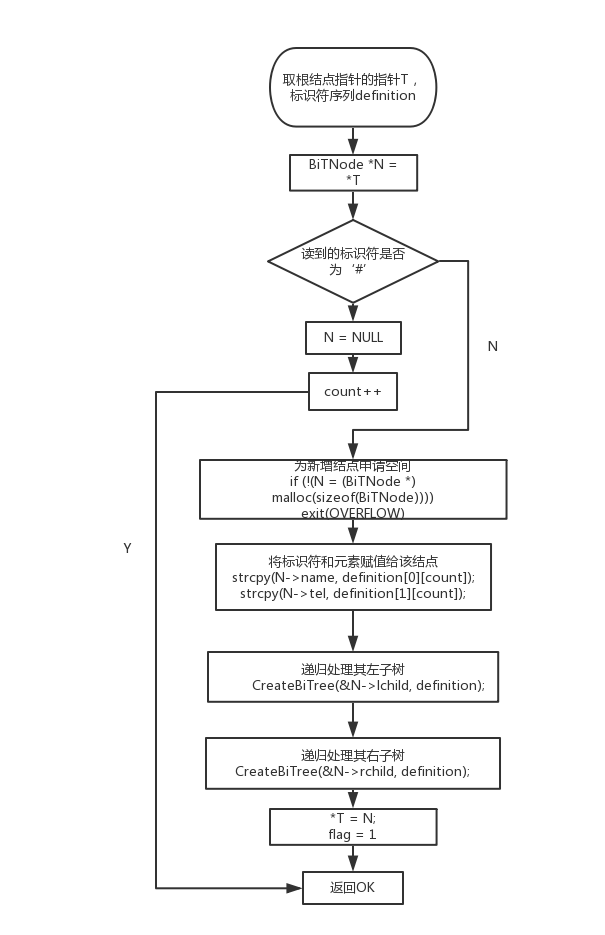
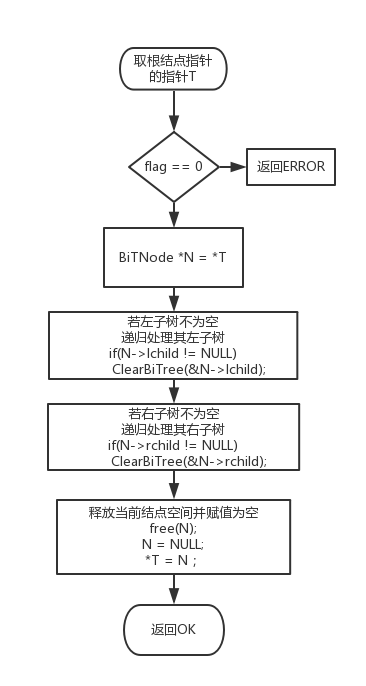
**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

图3-5创建二叉树算法流程图

**4.清空二叉树**

**算法输入：**根结点指针的指针

**算法输出：**实现状态（完成OK，未创建返回ERROR）

**算法处理步骤描述：**

（1）如果flag为0则二叉树未创建返回ERROR

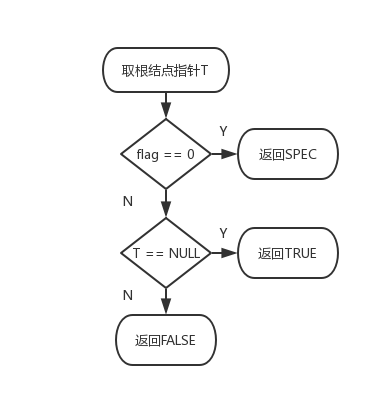
（2）如果左（右）子树不为空，则递归其左（右）子树

（3）释放当前结点，赋值为空，返回OK

如图3-6所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

图3-6清空二叉树算法流程图

**5.判定空二叉树**

**算法输入：**根结点指针

**算法输出：**实现状态（空TRUE，不为空FALSE，未创建SPEC）

**算法处理步骤描述：**

（1）如果flag为0则二叉树未创建返回ERROR

（2）如果根结点指针为空，则为空树，返回TRUE

（3）如果根结点指针不为空，则不为空树，返回FALSE

如图3-7所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

图3-7判定空二叉树算法流程图

**6.求二叉树深度**

**算法输入：**根结点指针

**算法输出：**二叉树的深度depth（空0，未创建SPEC）

**算法处理步骤描述：**

（1）如果flag为0则二叉树未创建返回SPEC

（2）如果根结点指针为空，则为空树，返回0

（3）递归该函数分别求出左右子树深度

（4）如果左子树深度大于右子树，则总深度为左子树深度加1，否则为右子树深度+1

（5）返回总深度depth

如图3-8所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

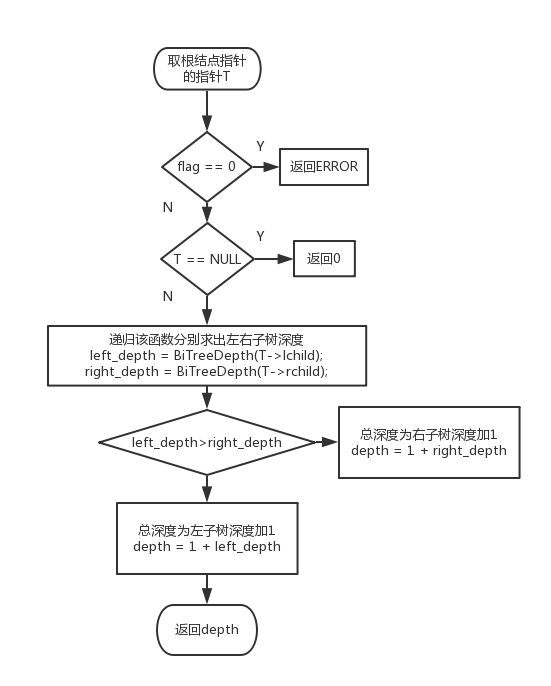
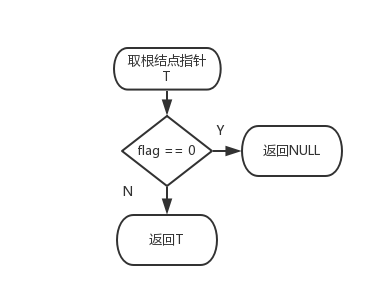


图3-8求二叉树深度算法流程图

**7.获得根结点**

**算法输入：**根结点指针

**算法输出：**BiTNode指针T（未创建NULL）

**算法处理步骤描述：**

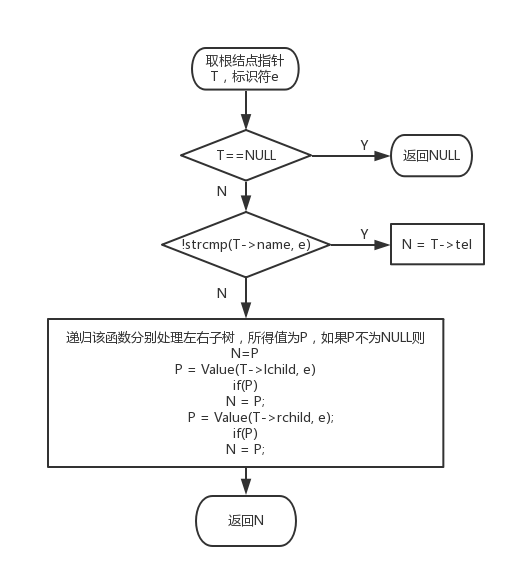
（1）如果flag为0则二叉树未创建返回ERROR

（2）返回T

如图3-9所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

图3-9获得根算法流程图



**8.获得结点**

**算法输入：**根结点指针，标识符

**算法输出：**元素值

**算法处理步骤描述：**

（1）如果该结点指针为空则返回NULL

（2）如果找到相应结点，用N暂存其值

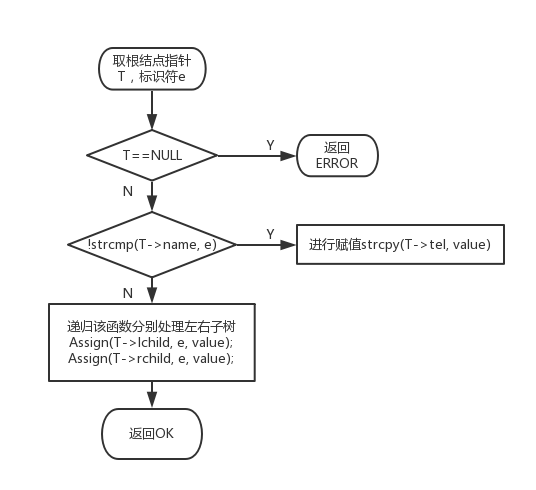
（3）递归该函数分别处理左右子树，所得值为P，如果P不为NULL则N=P

（4）返回N

如图3-10所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

图3-10获得结点算法流程图

**9.结点赋值**

**算法输入：**根结点指针，标识符，元素值

**算法输出：**实现状态（空ERROR，成功OK）

**算法处理步骤描述：**

（1）如果该结点指针为空则返回ERROR

（2）如果找到相应结点，进行赋值

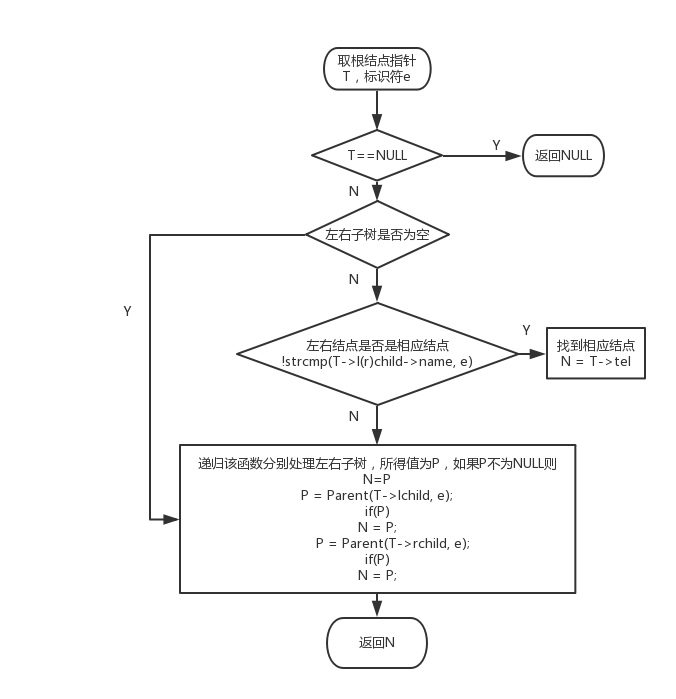
（3）递归该函数分别处理左右子树

（4）返回OK

如图3-11所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=0。

图3-11结点赋值算法流程图



**10.获得双亲结点**

**算法输入：**根结点指针，标识符

**算法输出：**BiTNode指针

**算法处理步骤描述：**

（1）如果该结点指针为空则返回NULL

（2）如果找到左右孩子为相应结点，则用N暂存该结地址

（3）递归该函数分别处理左右子树，所得值为P，如果P不为NULL则N=P

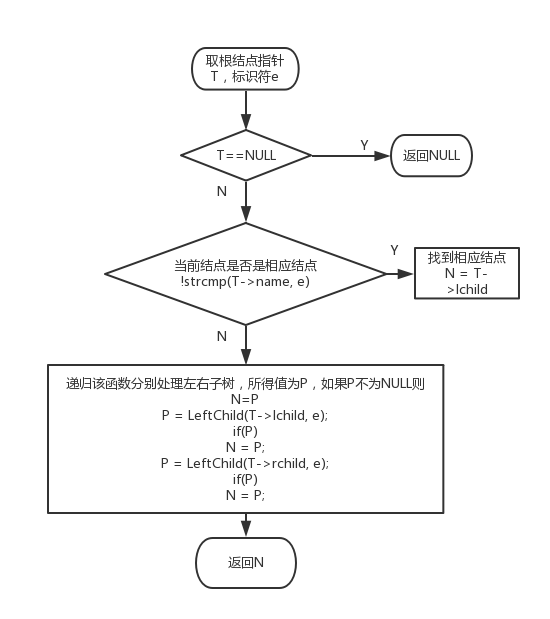
（4）返回N

如图3-12所示。

**算法时空复杂度分析：**

T(n)=O(n);S(n)=O(1)。

图3-12获得双亲结点算法流程图

**11.获得左孩子结点**

**算法输入：**根结点指针，标识符

**算法输出：**BiTNode指针

**算法处理步骤描述：**

（1）如果该结点指针为空则返回NULL

（2）如果找到相应结点，用N暂存其左孩子地址

（3）递归该函数分别处理左右子树，所得值为P，如果P不为NULL则更新N的值，N=P

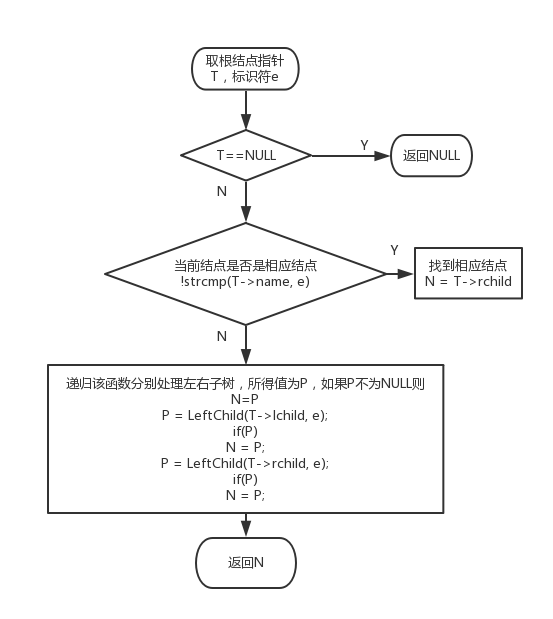
（4）返回N

如图3-13所示。

**算法时空复杂度分析：**

T(n)=O(n);S(n)=O(1)。

图3-13获得左孩子结点算法流程图

**12.获得右孩子结点**

**算法输入：**根结点指针，标识符

**算法输出：**BiTNode指针

**算法处理步骤描述：**

（1）如果该结点指针为空则返回NULL

（2）如果找到相应结点，用N暂存其右孩子地址

（3）递归该函数分别处理左右子树，所得值为P，如果P不为NULL则更新N的值，N=P

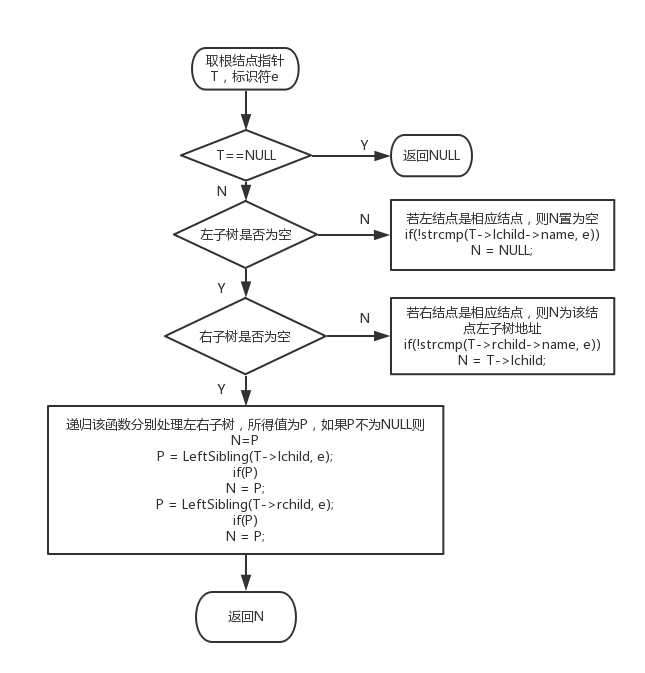
（4）返回N

如图3-14所示。

**算法时空复杂度分析：**

T(n)=O(n);S(n)=O(1)。

图3-14获得右孩子结点算法流程图

**13.获得左兄弟结点**

**算法输入：**根结点指针，标识符

**算法输出：**BiTNode指针

**算法处理步骤描述：**

（1）如果该结点指针为空则返回NULL

（2）如果左子树不为空，且左孩子结点为相应结点则将N置为NULL

（3）如果右子树不为空，且右孩子结点为相应结点则将N置

为该结点左孩子结点地址

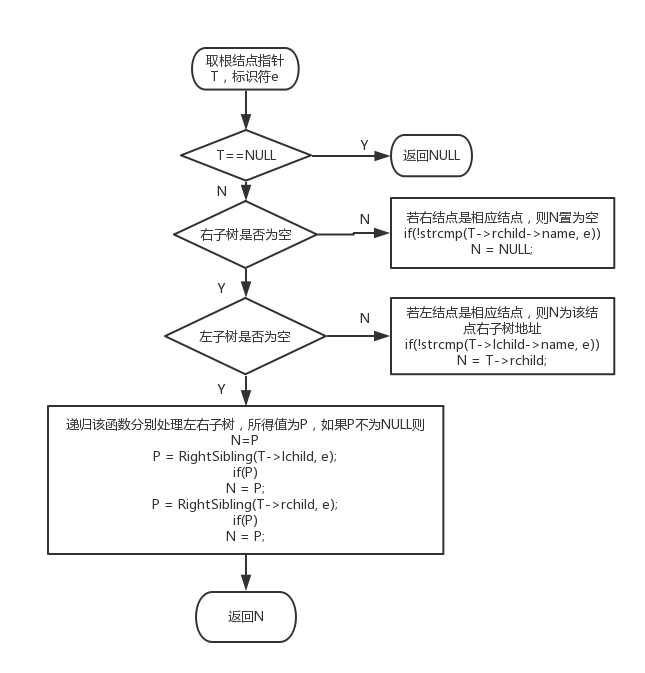
（4）递归该函数分别处理左

右子树，所得值为P，如果P不为NULL则更

新N的值，N=P

（5）返回N 图3-15获得左兄弟结点算法流程图

如图3-15所示。 **算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

**14.获得右兄弟结点**

**算法输入：**根结点指针，标识符

**算法输出：**BiTNode指针

**算法处理步骤描述：**

（1）如果该结点指针为空则返回NULL

（2）如果右子树不为空，且右孩子结点为相应结点则将N置为NULL

（3）如果左子树不为空，且左孩子结点为相应结点则将N置

为该结点右孩子结点地址

（4）递归该函数分别处理左

右子树，所得值为P，如果P不为NULL则更

新N的值，N=P

（5）返回N 图3-16获得右兄弟结点算法流程图

如图3-16所示。 **算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

**15.插入子树**

**算法输入：**根结点指针，要插入的结点的指针，标识（左或右子树），要插入的子树的根结点指针

**算法输出：**完成状态（未创建SPEC，插入成功OK）

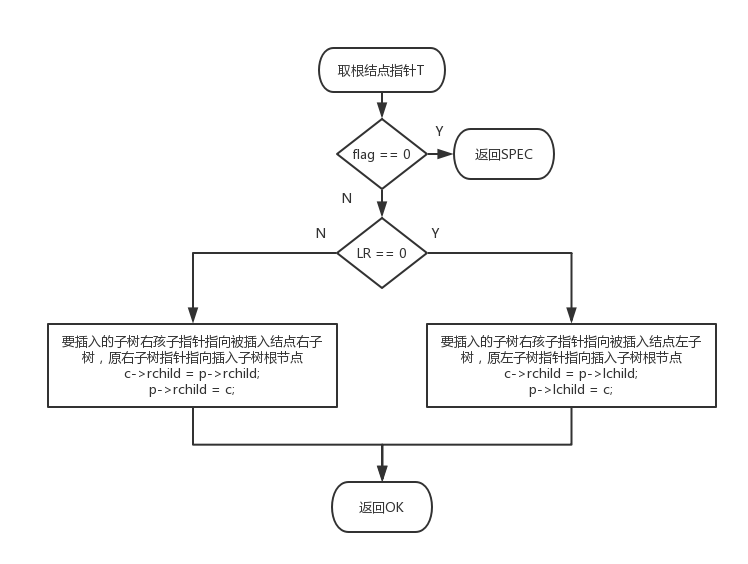
**算法处理步骤描述：**

（1）如果flag==0则二叉树未创建返回SPEC

（2）如果LR==0，则插入为结点左子树，要插入的子树右孩子指针指向被插入结点左子树，原左子树指针指向插入子树根节点

（3）如果LR==1，则插入为结点右子树，要插入的子树右孩子指针指向被插入结点右子树，原右子树指针指向插入子树根节点

（4）返回OK

如图3-17所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

图3-17插入子树算法流程图

**16.删除子树**

**算法输入：**根结点指针，要进行删除操作的结点的指针，标识（左或右子树）

**算法输出：**完成状态（删除失败SPEC，删除成功OK）

**算法处理步骤描述：**

1. 如果flag==0则二叉树未创建返回SPEC
2. 如果LR==0，则删除的为结点左子树，如果左孩子不为空则调用ClearBiTNode函数进行删除，否则输出左子树为空返回SPEC
3. 如果LR==1，则删除的为结点右子树，如果右孩子不为空则调用ClearBiTNode函数进行删除，否则输出右子树为空返回SPEC

（4）返回OK

如图3-18所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

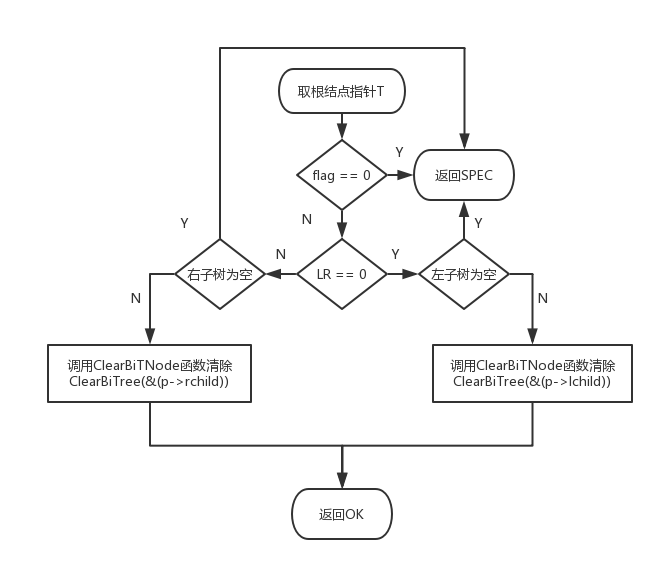
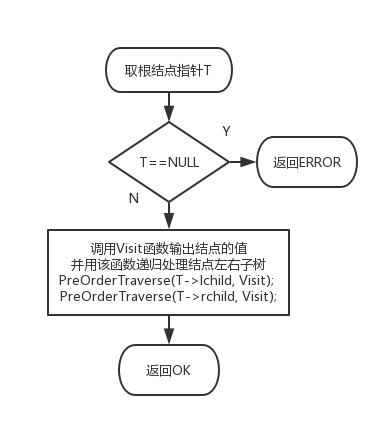


图3-18删除子树算法流程图

**17.先序遍历**

**算法输入：**根结点指针，Visit函数

**算法输出：**完成状态（二叉树为空ERROR，遍历成功OK）

**算法处理步骤描述：**

（1）如果根结点指针为空，则二叉树为空返回ERROR

（2）调用Visit函数输出结点的值

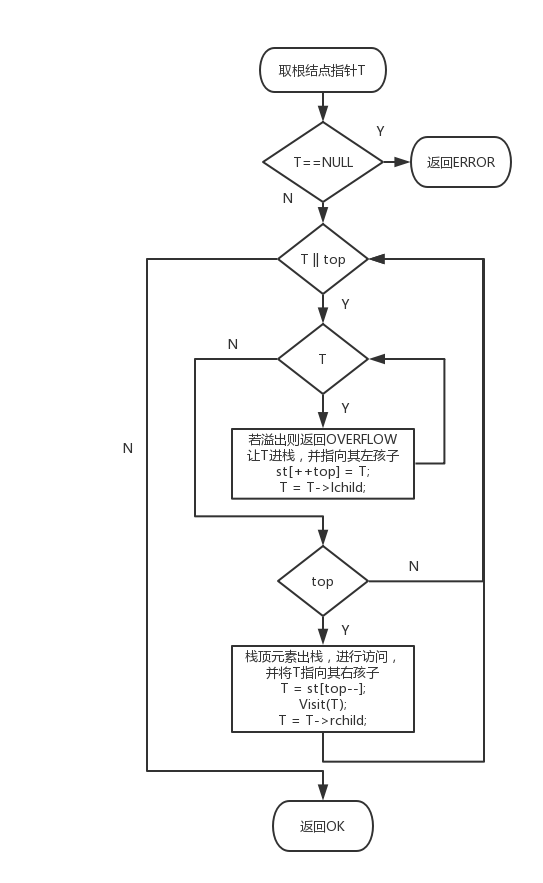
（3）用该函数递归处理结点左右子树

（4）返回OK

如图3-19所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=0。

图3-19先序遍历算法流程图



**18.中序遍历**

**算法输入：**根结点指针，Visit函数

**算法输出：**完成状态（二叉树为空ERROR，遍历成功OK）

**算法处理步骤描述：**

（1）如果根结点指针为空，则二叉树为空返回ERROR

（2）如果T不为0则让T进栈，并指向其左孩子

（3）如果栈中还有元素，则栈顶元素出栈，进行访问，并将T指向其右孩子

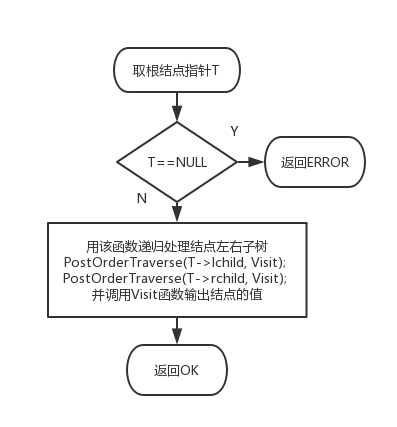
（4）重复（2），（3）直到T为空且栈中无元素

（5）返回OK

如图3-20所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(n)。

图3-20中序遍历算法流程图

**19.后序遍历**

**算法输入：**根结点指针，Visit函数

**算法输出：**完成状态（二叉树为空ERROR，遍历成功OK）

**算法处理步骤描述：**

（1）如果根结点指针为空，则二叉树为空返回ERROR

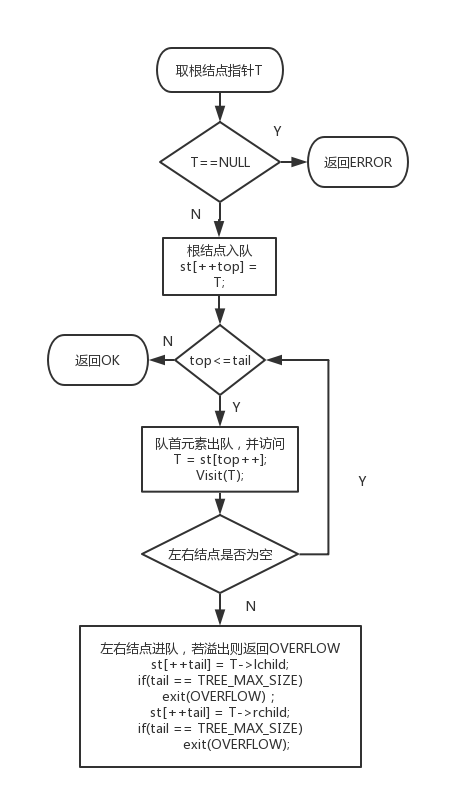
（2）用该函数递归处理结点左右子树

（3）调用Visit函数输出结点的值

（4）返回OK

如图3-21所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=0。

图3-21后序遍历算法流程图

**20.按层遍历**

**算法输入：**根结点指针，Visit函数

**算法输出：**完成状态（二叉树为空ERROR，遍历成功OK）

**算法处理步骤描述：**

（1）如果根结点指针为空，则二叉树为空返回ERROR

（2）根结点进队

（3）队首元素出队，并访问，如果其左右结点不为空，则进队，若溢出则返回OVERFLOW

（4）重复（3）直到队为空

（5）返回OK

如图3-22所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(n)。

图3-22按层遍历算法流程图

## 3.3 系统实现

## 程序开发程序为Dev-c，实现语言为C语言。

主要函数为：

status InitBiTree(BiTNode \*\*T);

status DestroyBiTree(BiTNode \*\*T);

Status CreateBiTree(BiTNode \*\*T, ElemType1

definition[2][TREE\_MAX\_SIZE][NAME\_MAX\_SIZE]);

status ClearBiTree(BiTNode \*\*T);

status BiTreeEmpty(BiTNode \*T);

int BiTreeDepth(BiTNode \*T);

BiTNode \*Root(BiTNode \*T);

ElemType2 \*Value(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

status Assign(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE], ElemType2 value[TEL\_MAX\_SIZE]);

BiTNode \*Parent(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*LeftChild(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*RightChild(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*LeftSibling(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*RightSibling(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

status InsertChild(BiTNode \*T, BiTNode \*p, int LR, BiTNode \*c);

status DeleteChild(BiTNode \*T, BiTNode \*p, int LR);

status PreOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

status InOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

status PostOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

status LevelOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

void Visit(BiTNode \*T);

BiTNode \*Find(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

status Output(BiTNode \*T, FILE \*fp);

status MenuInsert(Menu \*M, BiTNode \*T, char name[TREE\_NAME\_SIZE]);

status NewBiTree(BiTNode \*\*T);

int FindBiTree(Menu M, char name[TREE\_NAME\_SIZE]);

BiTNode \*ChooseBiTree(Menu M, char name[TREE\_NAME\_SIZE]);

status MenuDelete(Menu \*M, char name[TREE\_NAME\_SIZE]);

主函数中通过switch来实现输入不同的数字调用不同的功能并调用其对应的函数，基础功能对应一到二十。

本程序中包含文件的读写功能，对应于功能的二十一和二十二。

本程序包含多表操作功能，相应的函数有MenuInsert，MenuDelete，NewBiTree和ChooseBiTree，对应于功能的二十三到二十四。

**测试**

测试文件为：

Aa a1

Bb b2

Cc c3

#

#

Dd d4

#

#

Ee e5

#

Ff f6

#

#

为先序序列构成的二叉树

**示例1：**

对创建表功能的测试

测试用例及结果如表1-1所示

表3-1初始化二叉树测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 空间充足 | 创建成功  给出提示 | 8](161[28((IXQ[DGS8A}UA  符合理论结果 |

**示例2：**

对销毁二叉树功能的测试

测试用例及结果如表3-2所示

表3-2销毁二叉树测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 销毁失败给出提示 | DO1$CH(~`}F2)6VG%M58}QB  符合理论结果 |
| 用例二 | 已创建二叉树 | 销毁成功给出提示 | R[Z)0[HH_40K}98G9O)L8K7  符合理论结果 |

**示例3：**

对创建二叉树功能的测试

测试用例及结果如表3-3所示

表3-3创建表测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 空间充足 | 创建成功  给出提示 | 8RO4B`Z]MAZCP8I1Y{Y8WGI  符合理论结果 |
| 用例二 | 空间不足 | 创建失败  空间溢出  给出提示 |  |

**示例4：**

对清空二叉树功能的测试

测试用例及结果如表3-4所示

表3-4清空二叉树测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 二叉树未创建给出提示 | ZN5T]QLU76~B[036LUWI@O5  符合理论结果 |
| 用例二 | 测试用例 | 清除成功，给出提示 | ({UD4VY7VRFVFN6ZBVUMA}A  符合理论结果 |

**示例5：**

对判定空二叉树功能的测试

测试用例及结果如表3-5所示

表3-5判定空二叉树测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 判定失败给出提示 | ZN5T]QLU76~B[036LUWI@O5ZN5T]QLU76~B[036LUWI@O5  符合理论结果 |
| 用例二 | 空树 | 返回TRUE，给出提示 | }PI2$PC9``HXO7~LCW1_(IU  符合理论结果 |
| 用例三 | 测试用例 | 返回FALSE，给出提示 | 9[U][YZ2%Y804`4IWU%NPQW9B6E6IL3P)3O32`QSNEW1VH  符合理论结果 |

**示例6：**

对求二叉树深度功能的测试

测试用例及结果如表3-6所示

表3-6获求二叉树深度测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 判定失败给出提示 | ZN5T]QLU76~B[036LUWI@O5ZN5T]QLU76~B[036LUWI@O5  符合理论结果 |
| 用例二 | 空树 | 返回0，给出提示 | P1~EZ3[J21A95TNO_~XNDLF  符合理论结果 |
| 用例三 | 测试用例 | 返回二叉树深度，给出提示 | 9[U][YZ2%Y804`4IWU%NPQW]D`YH%MDN3ZDVKHLA5}W0$1  符合理论结果 |

**示例7：**

对获得根结点功能的测试

测试用例及结果如表3-7所示

表3-7获得根结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5ZN5T]QLU76~B[036LUWI@O5  符合理论结果 |
| 用例二 | 测试用例 | 返回根结点指针，给出提示 | R7L`42ABHQ)@[EE8)X85~CK  符合理论结果 |

**示例8：**

对获得结点功能的测试

测试用例及结果如表3-8所示

表3-8获得结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5ZN5T]QLU76~B[036LUWI@O5  符合理论结果 |
| 用例二 | 未找到相应结点 | 返回NULL，给出提示 | 1J2RI{1%_78NQ@~Q%0`I)R71J2RI{1%_78NQ@~Q%0`I)R7  符合理论结果 |
| 用例三 | 测试用例，Bb | 返回相应结点值，给出提示 | 9[U][YZ2%Y804`4IWU%NPQW$(H`_UNFL$]IJGRH~SBXZYY  符合理论结果 |

**示例9：**

对结点赋值功能的测试

测试用例及结果如表3-9所示

表3-9结点赋值测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5ZN5T]QLU76~B[036LUWI@O5  符合理论结果 |
| 用例二 | 未找到相应结点 | 未找到相应结点，给出提示 | 1J2RI{1%_78NQ@~Q%0`I)R7N`X6A%Y}F1OMNQ9UGJ(_B96  符合理论结果 |
| 用例三 | 测试用例，Bb | 赋值成功，给出提示 | 9[U][YZ2%Y804`4IWU%NPQW~58[WK~%HDHRK%@@5AOD_{D  符合理论结果 |

**示例10：**

对获得双亲结点功能的测试

测试用例及结果如表3-10所示

表3-10获得双亲结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或未找到相应结点 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5AXJQC6I{MGM7P_5~1)JB)0N  符合理论结果 |
| 用例二 | 测试用例，Bb | 返回双亲结点地址，给出提示 | 9[U][YZ2%Y804`4IWU%NPQWY`942M9U~80(BI@5D(T3A`F  符合理论结果 |

**示例11：**

对获得左孩子结点功能的测试

测试用例及结果如表3-11所示

表3-11获得左孩子结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或未找到相应结点 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5OLQ5Q4JF_Y_XIY6V@@V9UKV  符合理论结果 |
| 用例二 | 测试用例，Bb | 返回左孩子地址，给出提示 | 9[U][YZ2%Y804`4IWU%NPQW}IPBIING`OTC3)@UJF~{FCF  符合理论结果 |

**示例12：**

对获得右孩子结点功能的测试

测试用例及结果如表3-12所示

表3-12获得右孩子结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或未找到相应结点 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5NL3SF01SB7KOC[[KL_KI[UN  符合理论结果 |
| 用例二 | 测试用例，Aa | 返回左孩子地址，给出提示 | 9[U][YZ2%Y804`4IWU%NPQW}{)(SGVG3[BZTH3)4)2817R  符合理论结果 |

**示例13：**

对获得左兄弟结点功能的测试

测试用例及结果如表3-13所示

表3-13获得左兄弟结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或未找到相应结点 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5~GARDKE0~XCR6B6H[30WNGX  符合理论结果 |
| 用例二 | 测试用例，Dd | 返回左兄弟地址，给出提示 | 9[U][YZ2%Y804`4IWU%NPQWV19FP(U3TM)()FOM0X{R1BK  符合理论结果 |

**示例14：**

对获得右兄弟结点功能的测试

测试用例及结果如表3-14所示

表3-14获得右兄弟结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或未找到相应结点 | 获取失败给出提示 | ZN5T]QLU76~B[036LUWI@O5]WS8LI29WN)Q$4}]{J3YNBB  符合理论结果 |
| 用例二 | 测试用例，Dd | 返回右兄弟地址，给出提示 | 9[U][YZ2%Y804`4IWU%NPQW8F8Y%{9[)A1{9CRQ4S`64`9  符合理论结果 |

**示例15：**

对插入子树功能的测试

测试用例及结果如表3-15所示

表3-15插入子树测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 插入失败，给出提示 | ZN5T]QLU76~B[036LUWI@O5(37]K6PZ([H[F5)QW8$I{S2  符合理论结果 |
| 用例二 | 测试用例，Bb，0,1，Rr，r7 | 插入成功，给出提示 | 9[U][YZ2%Y804`4IWU%NPQWHYAYC3B79_1~13YLWWR6RD2  符合理论结果 |

**示例16：**

对删除子树功能的测试

测试用例及结果如表3-16所示

表3-16删除子树测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树 | 删除失败，给出提示 | ZN5T]QLU76~B[036LUWI@O5_S0TSYNI1]$X5PFMNZ_495A  符合理论结果 |
| 用例二 | 测试用例，Bb，0 | 删除成功，给出提示 | 9[U][YZ2%Y804`4IWU%NPQWITI]]~5YAT$IXS$R~]HAIXK  符合理论结果 |

**示例17：**

对先序遍历功能的测试

测试用例及结果如表3-17所示

表3-17先序遍历测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或为空 | 删除失败，给出提示 | ZN5T]QLU76~B[036LUWI@O5@@Z{DB[CSONU}NYBZ[QV[[Y  符合理论结果 |
| 用例二 | 测试用例 | 显示先序遍历结果 | 9[U][YZ2%Y804`4IWU%NPQWF@G`[IN9RAD)NYUX4WRL@87F@G`[IN9RAD)NYUX4WRL@87  符合理论结果 |

**示例18：**

对中序遍历功能的测试

测试用例及结果如表3-18所示

表3-18中序遍历测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或为空 | 删除失败，给出提示 | ZN5T]QLU76~B[036LUWI@O5@@Z{DB[CSONU}NYBZ[QV[[Y  符合理论结果 |
| 用例二 | 测试用例 | 显示中序遍历结果 | 9[U][YZ2%Y804`4IWU%NPQWF@G`[IN9RAD)NYUX4WRL@87X7Y[ROZ2P}0@(%OU{ZPTQZ9  符合理论结果 |

**示例19：**

对后序遍历功能的测试

测试用例及结果如表3-19所示

表3-19后序遍历测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或为空 | 删除失败，给出提示 | ZN5T]QLU76~B[036LUWI@O5@@Z{DB[CSONU}NYBZ[QV[[Y  符合理论结果 |
| 用例二 | 测试用例 | 显示后序遍历结果 | 9[U][YZ2%Y804`4IWU%NPQWF@G`[IN9RAD)NYUX4WRL@87ETO{BE0%4PL1YOL6L[TL]`9  符合理论结果 |

**示例20：**

对按层遍历功能的测试

测试用例及结果如表3-20所示

表3-20按层遍历测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建二叉树或为空 | 删除失败，给出提示 | ZN5T]QLU76~B[036LUWI@O5@@Z{DB[CSONU}NYBZ[QV[[Y  符合理论结果 |
| 用例二 | 测试用例 | 显示按层遍历结果 | 9[U][YZ2%Y804`4IWU%NPQWF@G`[IN9RAD)NYUX4WRL@8747VQ$MQ318__OXW14B6GS6E  符合理论结果 |

经过实验，程序整体测试效果良好，程序基本满足实验要求。

## 3.4 实验小结

本次实验是第一次创建二叉树，且这是第一次进行非线性结构的创建，难度相比于前两次的实验可以说是有很大的提升，在实验过程中也遇到了很多以前没有出现过的问题，特别是以前没有太多练习过的递归算法，而在这次实验中却要大量使用，导致在实验过程中出现了很多难以解决的问题，不过最终还是克服了，通过这次的实验，我更加熟悉了二叉树这种数据结构的特点和基本的使用操作方法，也更加熟悉了不同的算法，对于不同的算法在各个场合的使用也有了更深的理解，在以后的实验和编程过程中我会更多地去尝试使用非线性结构去解决问题，以更加熟练掌握这种结构，最终能够灵活应用。

# 

# 4基于邻接表的图实现

## 4.1 问题描述

采用邻接表作为无向网的物理结构，用函数实现13个基本功能，并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。演示系统可选择实现无向网的文件形式保存及读取和多个无向网的管理。

## 4.1.1 实验目的

通过实验达到

⑴加深对图的概念、基本运算的理解；

⑵熟练掌握图的逻辑结构与物理结构的关系；

⑶以邻接表作为物理结构，熟练掌握图基本运算的实现。

## 4.1.2 实验中需完成的函数

依据最小完备性和常用性相结合的原则，以函数形式定义了创建图、销毁图、查找顶点、获得顶点值和顶点赋值等13种基本运算，具体运算功能定义如下。

⑴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。

⑵销毁图：树函数名称是DestroyBiTree(T)；初始条件图G已存在；操作结果是销毁图G。

⑶查找顶点：函数名称是LocateVex(G,u)；初始条件是图G存在，u和G中的顶点具有相同特征；操作结果是若u在图G中存在，返回顶点u的位置信息，否则返回其它信息。

⑷获得顶点值：函数名称是GetVex (G,v)；初始条件是图G存在，v是G中的某个顶点；操作结果是返回v的值。

⑸顶点赋值：函数名称是PutVex (G,v,value)；初始条件是图G存在，v是G中的某个顶点；操作结果是对v赋值value。

⑹获得第一邻接点：函数名称是FirstAdjVex(&G, v)；初始条件是图G存在，v是G的一个顶点；操作结果是返回v的第一个邻接顶点，如果v没有邻接顶点，返回“空”。

⑺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是G的一个顶点,w是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，如果w是最后一个邻接顶点，返回“空”。

⑻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。

⑼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。

⑽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>。

⑾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>。

⑿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。

⒀广度优先搜索遍历：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。

## 4.1.3 需求分析

在现实生活中很多复杂的关系都可以用图来描述，很多实际的具体问题都能抽象成图进行计算操作并利用所构建的图去解决一些问题，例如常见的用prim算法或kruskal算法对一个图的最小生成树进行求解，从而解决最优化等方面的问题

## 4.2 系统设计

**4.2.1 总体设计**

用函数实现创建图、销毁图、查找顶点、获得顶点值和顶点赋值等13种基本运算并用函数进行封装，在主程序中设置菜单显示各个功能并进行标号，通过switch函数对前面运算函数进行调用以实现对应标号的功能。同时本程序中添加了文件存储和读取作为标号14，15对应的功能，也填加了图的插入，图的删除，创建新的图，选择图等功能实现多个图的管理，作为16到19号功能。

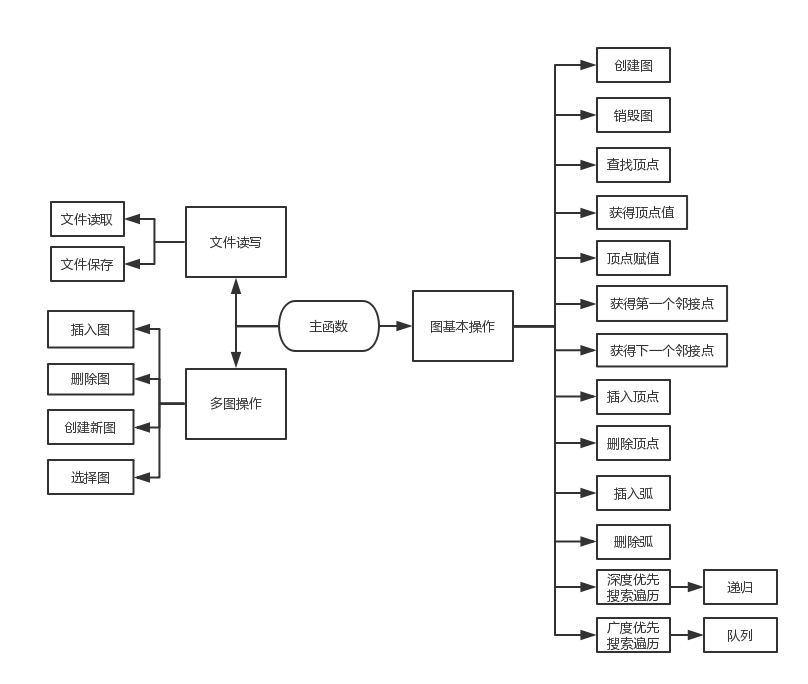


图4-1模块结构图

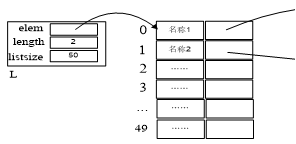
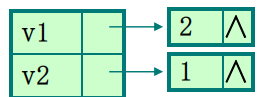
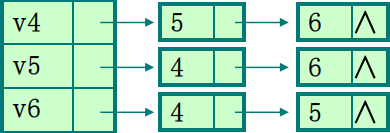


图4-2 多图管理的物理结构示意图

### 4.2.2 数据结构设计

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define SPEC -2

#define NONE -1

typedef int status;

typedef int ElemType\_Arc;

typedef int ElemType\_Ver;

#define MAX\_VERTEX\_NUM 20

#define MAX\_ARC\_NUM 200

#define MAX\_ARC\_NAME 20

#define MAX\_VERTEX\_NAME 20

#define GRAPH\_NAME\_SIZE 20

#define MENU\_INIT\_SIZE 50

int visited[50];

static char menu\_name[GRAPH\_NAME\_SIZE];

typedef struct ArcInfo{

char arc\_name[MAX\_ARC\_NAME];

ElemType\_Arc arc\_info;

}ArcInfo;

typedef struct VInfo{

char ver\_name[MAX\_VERTEX\_NAME];

ElemType\_Ver ver\_info;

}VInfo;

typedef struct ArcNode{//表结点结构类型

int adjvex; //该弧(边)的终点位置

struct ArcNode \*nextarc; //指向下一条弧的指针

ArcInfo \*info; //该弧的相关信息的指针

} ArcNode;

typedef struct VNode {//头结点的类型

VInfo data; //顶点信息

ArcNode \*firstarc; //指向第一条弧

}VNode, AdjList;

typedef struct {//邻接表

AdjList \*vertices;

int vexnum, arcnum; //图中顶点数n和边数e

int kind; //图的类型

} ALGraph;

typedef struct {

int adjvex\_start;

int adjvex\_end;

ArcInfo arc\_data;

}ArcCreateInfo;

typedef struct MenuName

{

char name[GRAPH\_NAME\_SIZE];

ALGraph graph;

}MenuName;

typedef struct Menu{

MenuName \*menuname;

int length;

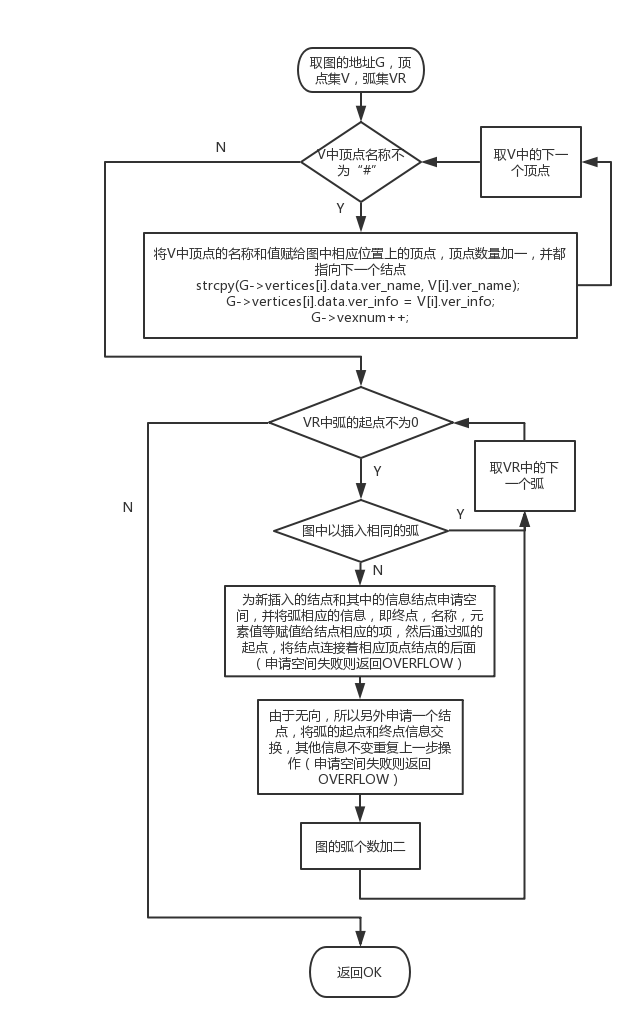
int menusize;

}Menu;

**4.2.3 算法设计**

**1.创建图**

**算法输入：**图指针，顶点集，弧集



**算法输出：**实现状态（完成OK，空间溢出OVERFLOW）

**算法处理步骤描述：**

（1）将V中顶点的名称和值赋给图中相应位置上的顶点，顶点数量加一，并都指向下一个结点，直到当前V中顶点名称为“#”

（2）若图中没有插入相同的弧，则为新插入的结点和其中的信息结点申请空间，并将弧相应的信息，即终点，名称，元素值等赋值给结点相应的项，然后通过弧的起点，将结点连接着相应顶点结点的后面（申请空间失败则返回OVERFLOW）

（3）由于无向，所以另外申请一个结点，将弧的起点和终点信息交换，其他信息不变重复上一步操作（申请空间失败则返回OVERFLOW）

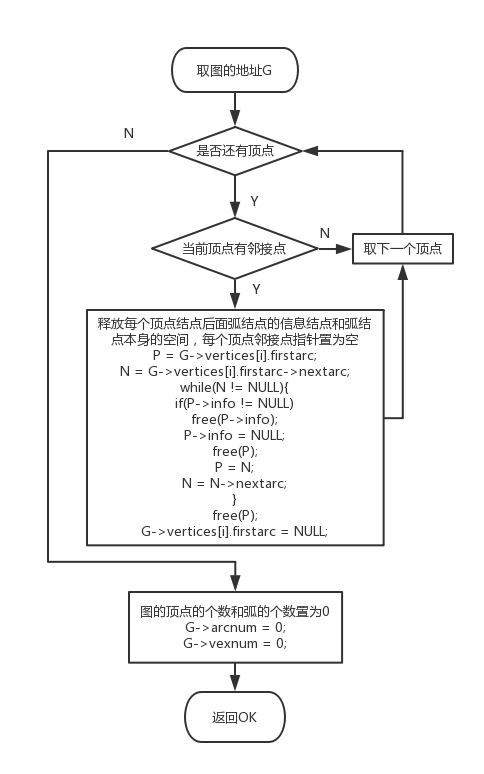
（4）图中弧的个数加二，取VR中的下一个弧，重复（2）（3）直到弧起点为0

（5）返回OK

如图4-3所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(n)。

图4-3创建图算法流程图

**2.销毁图**

**算法输入：**图指针

**算法输出：**实现状态（完成OK，未创建返回ERROR）

**算法处理步骤描述：**

（1）释放每个顶点结点后面弧结点的信息结点和弧结点本身的空间

（2）每个顶点邻接点指针置为空

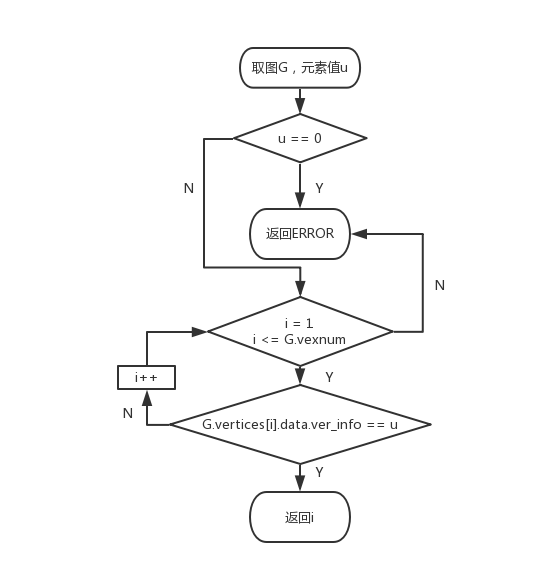
（3）图的顶点的个数和弧的个数置为0

（4）返回OK

如图4-4所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

图4-4销毁图算法流程图



**3.查找顶点**

**算法输入：**图，元素值

**算法输出：**顶点位置（未找到ERROR）

**算法处理步骤描述：**

（1）如果元素值为0，返回ERROR

（2）循环遍历所有顶点，若找到元素值则返回顶点位置

（3）未找到，返回ERROR

如图4-5所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

图4-5查找顶点算法流程图

**4.获得顶点**



**算法输入：**图，顶点位置

**算法输出：**顶点值（顶点位置错误或顶点值为0，ERROR）

**算法处理步骤描述：**

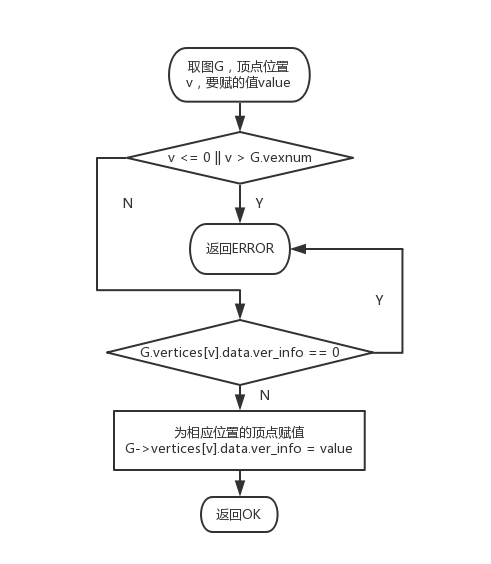
（1）如果顶点位置小于等于0或大于图中顶点数，或顶点值等于0，返回ERROR

（2）返回相应顶点的值

如图4-6所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

图4-6获得顶点算法流程图

****

**5.顶点赋值**

**算法输入：**图，顶点位置，要赋的值

**算法输出：**实现状态（成功OK，顶点位置错误或顶点值为

0，ERROR）

**算法处理步骤描述：**

（1）如果顶点位置小于等于0或大于图中顶点数，或顶点值等于0，返回ERROR

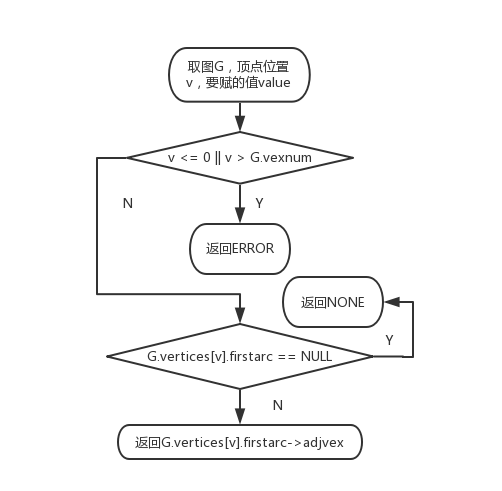
（2）为相应位置的顶点赋值

（3）返回OK

如图4-7所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

图4-7顶点赋值算法流程图

**6.获得第一邻接点**

**算法输入：**图，顶点位置

**算法输出：**第一邻接点位置（位置错误ERROR，无邻接

点NONE）

**算法处理步骤描述：**

1. 如果顶点位置小于等于0或大于图中顶点

数，返回ERROR

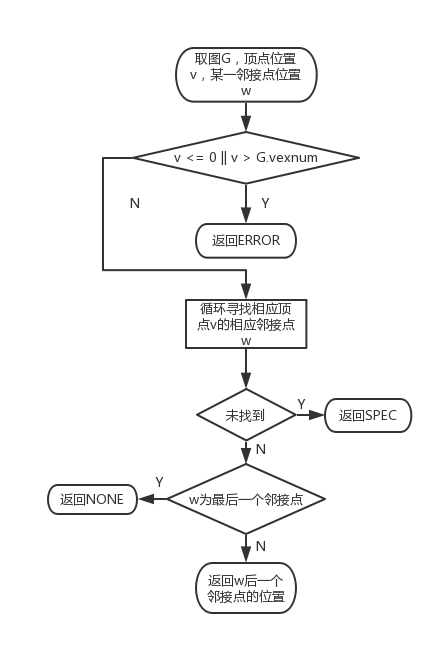
（2）如果该顶点无邻接点，返回NONE

（3）返回第一邻接点位置

如图4-8所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=0。

图4-8获得第一邻接点算法流程图

**7.获得下一邻接点**

**算法输入：**图，顶点位置，某一邻接点

**算法输出：**下一邻接点位置（位置错误ERROR，未

找到SPEC，后续无邻接点NONE）

**算法处理步骤描述：**

（1）如果顶点位置小于等于0或大于图中顶点

数，返回ERROR

（2）循环寻找相应顶点的相应邻接点，若未找到，返回

SPEC

（3）若找到，但为最后一个邻接点，则返回NONE

（4）若不为最后一个邻接点，则返回下一

个邻接点的位置

如图4-9所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

图4-9获得下一邻接点算法流程图

**8.插入顶点**

**算法输入：**图指针，顶点

**算法输出：**完成状态（插入成功OK，空间不足ERROR）

**算法处理步骤描述：**

（1）如果顶点数量大于等于最大顶点数量，则为顶点表新增空间，申请失败返回ERROR

（2）通过传入函数的顶点信息为相应新增位置上的顶点赋值

（3）返回OK

如图4-10所示。

**算法时空复杂度分析：**T(n)=O(1);S(n)=O(1)。

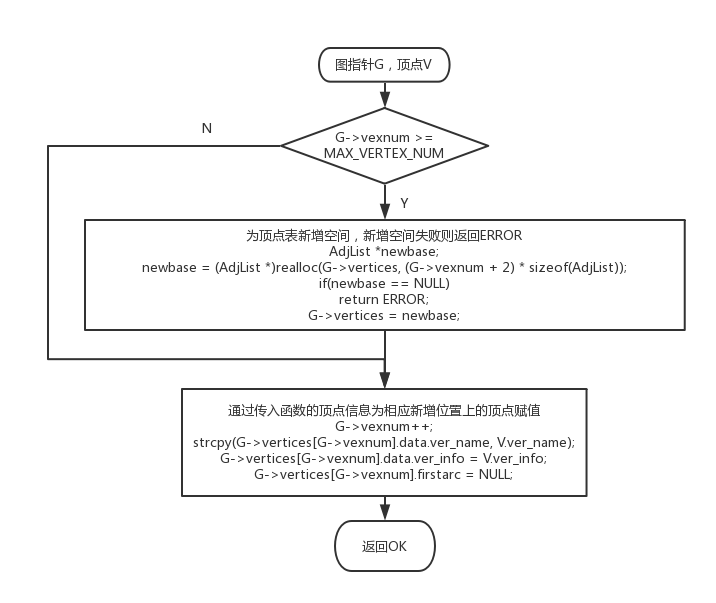


图4-10插入顶点算法流程图

**9.删除顶点**

**算法输入：**图指针，顶点位置

**算法输出：**实现状态（失败ERROR，成功OK）

**算法处理步骤描述：**

（1）如果顶点位置小于等于0或大于图中顶点数，返回ERROR

（2）如果相应顶点有邻接点，则释放所有弧结点及其信息结点的空间

（3）将该顶点结点邻接点置为空，元素值置为0，名称置为“#”

（4）遍历所有顶点，遍历所有有邻接点的顶点，若发现有邻接点为被删除的顶点的弧结点，则删除该结点。

（5）返回OK

如图4-11所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

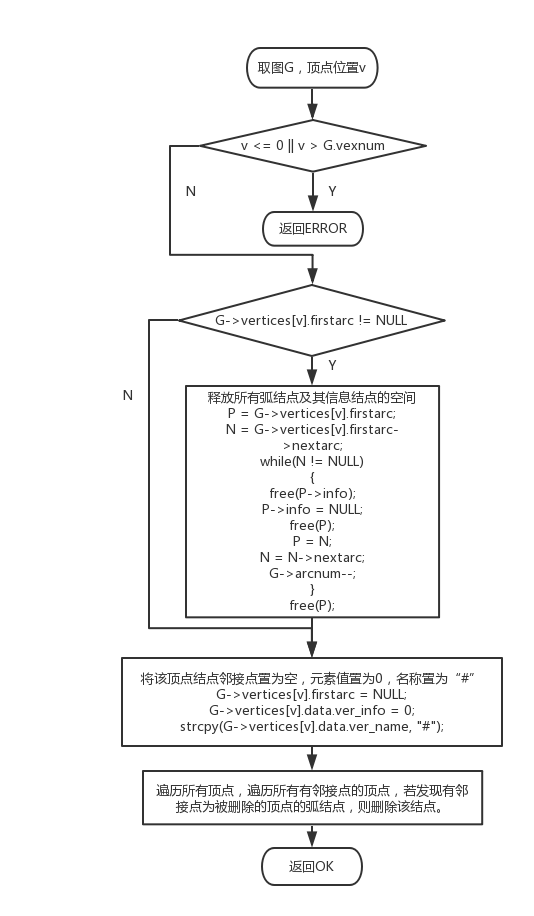
****

图4-11删除顶点算法流程图

**10.插入弧**

**算法输入：**图，弧的起点，弧的终点

**算法输出：**完成状态（插入成功OK，插入失败ERROR，空间不足OVERFLOW）

**算法处理步骤描述：**

（1）如果起点或终点位置小于等于0或大于图中顶点数，返回ERROR

（2）为新插入的弧结点和其中的信息结点申请空间，将信息输入至新插入结点，根据起点，将新增结点插入到相应顶点结点后。（申请空间失败返回OVERFLOW）

（3）因为是无向网，所以另申请空间，将起点和终点交换重复（2）的操作

（4）图的弧结点数量加二，返回OK

如图4-12所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

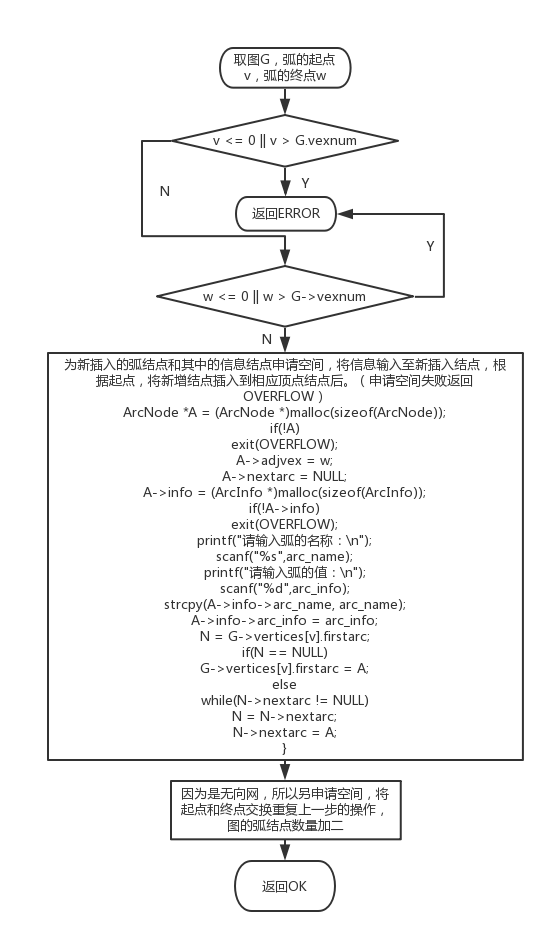


图4-12插入弧结点算法流程图

**11.删除弧**

**算法输入：**图，弧的起点，弧的终点

**算法输出：**完成状态（删除成功OK，删除失败ERROR）

**算法处理步骤描述：**

（1）如果未在相应的顶点v后找到了相应的邻接点w，则返回ERROR

（2）如果找到，则删除该弧结点，图的弧结点个数减一

（3）由于是无向网，所以将起点和终点信息调换，找到调换后相应弧结点进行删除，图的弧结点个数减一

（4）返回OK

如图4-13所示。

**算法时空复杂度分析：**T(n)=O(n);S(n)=O(1)。

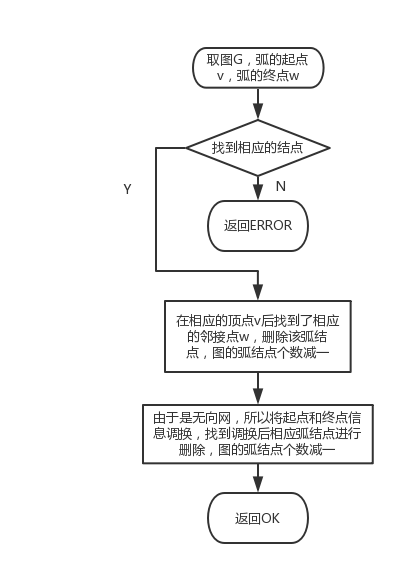


图4-13删除弧结点算法流程图

**12.深度优先搜索遍历**

**算法输入：**图，输出函数

**算法输出：**无

**算法处理步骤描述：**

（1）初始化辅助数组，记录已经遍历过的顶点

（2）遍历顶点

（3）若顶点未被遍历，则进行输出，相应改变辅助数组的值,在遍历顶点过程中，遍历其邻接点

（4）递归操作（3），每次将刚遍历的结点所指向的顶点作为新的起点

如图4-14所示。

**算法时空复杂度分析：**

T(n)=O(n+e);S(n)=O(n)。

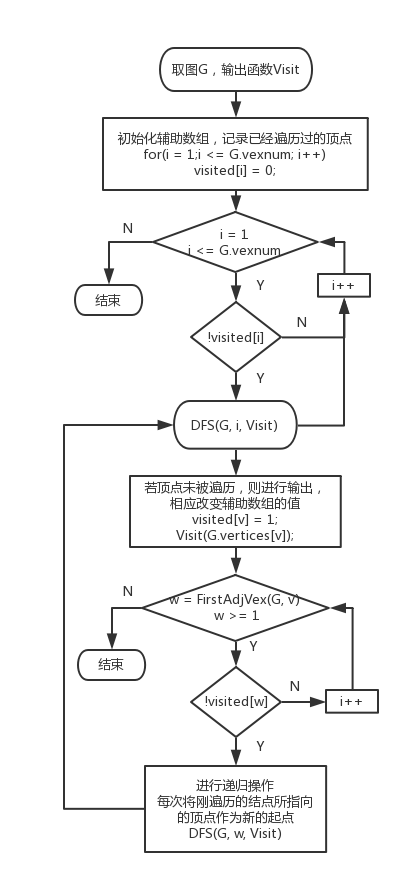


图4-14深度优先搜索遍历算法流程图

**13.广度优先搜索遍历**

**算法输入：**图，输出函数

**算法输出：**无

**算法处理步骤描述：**

（1）初始化辅助数组，记录已经遍历过的顶点

（2）创建队列数组

（3）遍历顶点，若已访问过，则继续访问下一个顶点

（4）未访问过的顶点访问，标记并进队

（5）队首元素出队，依次访问其邻接点，标记并入队

（6）重复（5）直到队为空

（7）访问下一个顶点，重复（3）（4）（5）（6）直到所有顶点都被访问

如图4-15所示。

**算法时空复杂度分析：**T(n)=O(e);S(n)=O(n)。

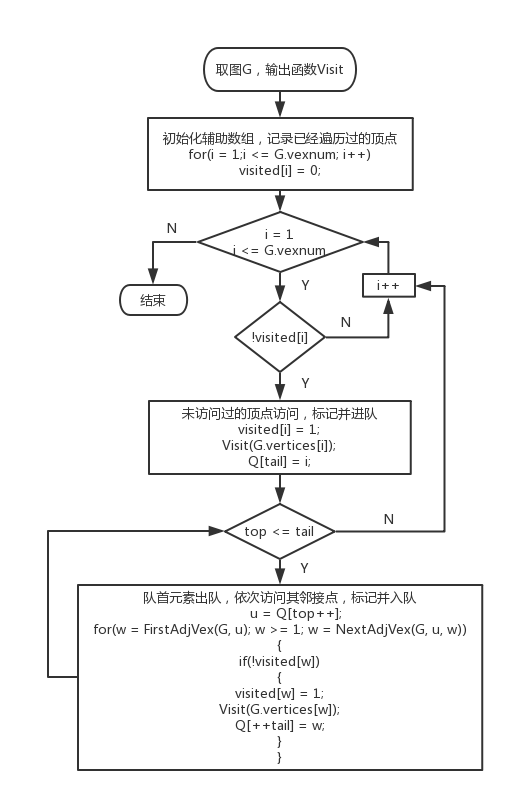


图4-15广度优先搜索遍历算法流程图

## 4.3 系统实现

## 程序开发程序为Dev-c，实现语言为C语言。

主要函数为：

status CreateGraph(ALGraph \*G,VInfo \*V,ArcCreateInfo \*VR);

status DestroyGraph(ALGraph \*G);

int LocateVex(ALGraph G, int u);

ElemType\_Ver GetVex(ALGraph G, int v);

status PutVex(ALGraph \*G, int v, ElemType\_Ver value);

int FirstAdjVex(ALGraph G, int v);

int NextAdjVex(ALGraph G, int v, int w);

status InsertVex(ALGraph \*G, VInfo V);

status DeleteVex(ALGraph \*G, int v);

status InsertArc(ALGraph \*G, int v, int w);

status DeleteArc(ALGraph \*G, int v, int w);

void DFSTraverse(ALGraph G, void (\*Visit) (VNode V));

void DFS(ALGraph G, int v, void (\*Visit) (VNode V));

void BFSTraverse(ALGraph G, void (\*Visit) (VNode V));

void Visit(VNode V);

status FindArcNode(ALGraph G, int v, int w);

status MenuInsert(Menu \*M, ALGraph G, char name[GRAPH\_NAME\_SIZE]);

status NewGraph(ALGraph \*G);

int FindGraph(Menu M, char name[GRAPH\_NAME\_SIZE]);

ALGraph ChooseGraph(Menu M, char name[GRAPH\_NAME\_SIZE]);

status MenuDelete(Menu \*M, char name[GRAPH\_NAME\_SIZE]);status CreateGraph(ALGraph \*G,VInfo \*V,ArcCreateInfo \*VR);

status DestroyGraph(ALGraph \*G);

int LocateVex(ALGraph G, int u);

ElemType\_Ver GetVex(ALGraph G, int v);

status PutVex(ALGraph \*G, int v, ElemType\_Ver value);

int FirstAdjVex(ALGraph G, int v);

int NextAdjVex(ALGraph G, int v, int w);

status InsertVex(ALGraph \*G, VInfo V);

status DeleteVex(ALGraph \*G, int v);

status InsertArc(ALGraph \*G, int v, int w);

status DeleteArc(ALGraph \*G, int v, int w);

void DFSTraverse(ALGraph G, void (\*Visit) (VNode V));

void DFS(ALGraph G, int v, void (\*Visit) (VNode V));

void BFSTraverse(ALGraph G, void (\*Visit) (VNode V));

void Visit(VNode V);

status FindArcNode(ALGraph G, int v, int w);

status MenuInsert(Menu \*M, ALGraph G, char name[GRAPH\_NAME\_SIZE]);

status NewGraph(ALGraph \*G);

int FindGraph(Menu M, char name[GRAPH\_NAME\_SIZE]);

ALGraph ChooseGraph(Menu M, char name[GRAPH\_NAME\_SIZE]);

status MenuDelete(Menu \*M, char name[GRAPH\_NAME\_SIZE]);

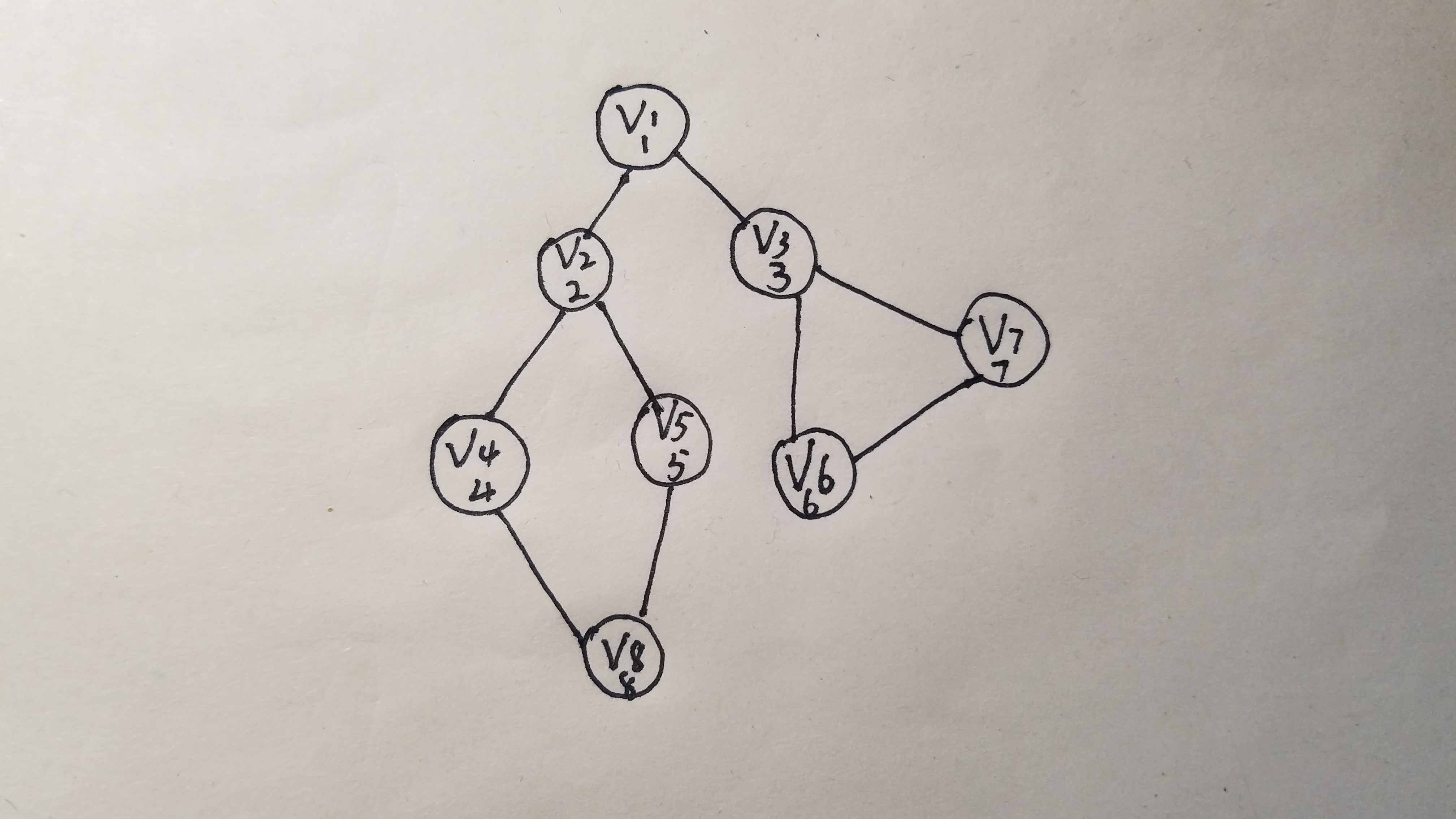
主函数中通过switch来实现输入不同的数字调用不同的功能并调用其对应的函数，基础功能对应一到十三。

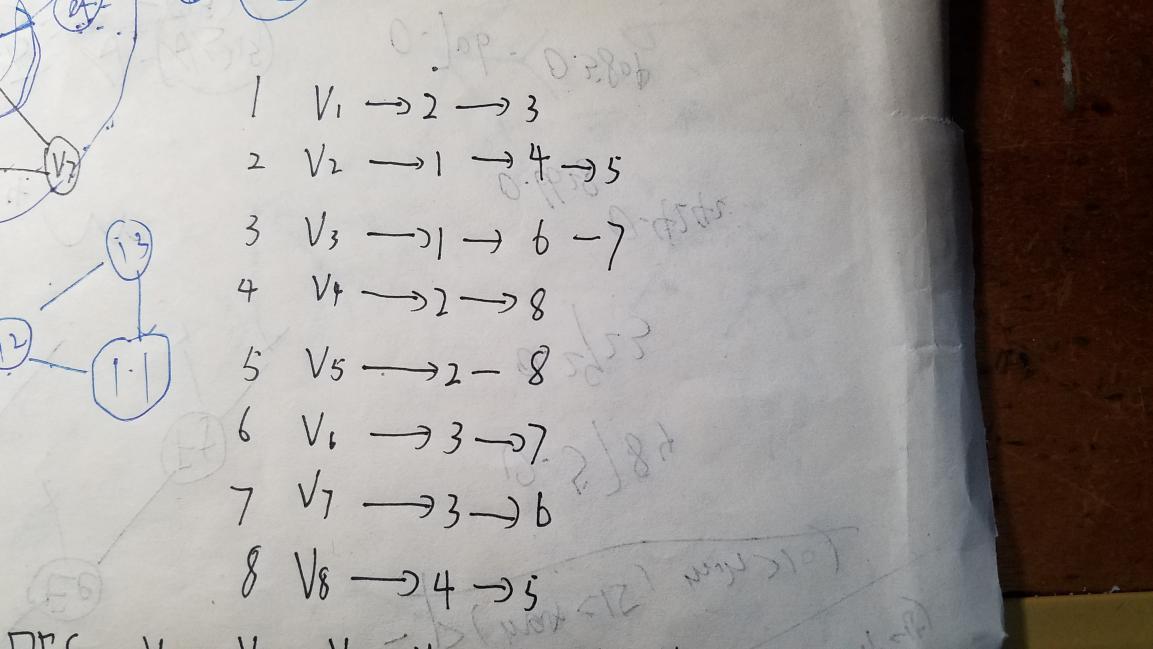
本程序中包含文件的读写功能，对应于功能的十四和十五。

本程序包含多表操作功能，相应的函数有MenuInsert，MenuDelete，NewBiTree和ChooseBiTree，对应于功能的十六到十九。

**测试**

测试文件为：



****

**图的构造及邻接表如上图所示**

**示例1：**

对创建图功能的测试

测试用例及结果如表4-1所示

表4-1创建图测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 空间充足 | 创建成功  给出提示 | Y9V$O6SYR]P5H[$JT9B9JKN  符合理论结果 |
| 用例二 | 空间不足 | 创建失败  空间溢出  给出提示 |  |

**示例2：**

对销毁图功能的测试

测试用例及结果如表4-2所示

表4-2销毁图测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 销毁失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 已创建图 | 销毁成功给出提示 | GTV~DN`]H1A@U7V~U%ZR%IC  符合理论结果 |

**示例3：**

对查找顶点功能的测试

测试用例及结果如表4-3所示

表4-3查找顶点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 查找失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，0 | 查找失败给出提示 | {9$829A$_LBJNS9$BL})S6N  符合理论结果 |
| 用例三 | 测试文件，3 | 查找成功，3 | {J`2FIJK@H(E8T}I{X]X6BW  符合理论结果 |

**示例4：**

对获得顶点值功能的测试

测试用例及结果如表4-4所示

表4-4获得顶点值测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 获取失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，0 | 获取失败给出提示 | J1L@1EA6V4@%R}6`~5((N~5  符合理论结果 |
| 用例三 | 测试文件，4 | 获取成功，4 | X6N[R~1`Y`~BE)K%$K8Z{_0  符合理论结果 |

**示例5：**

对顶点赋值功能的测试

测试用例及结果如表4-5所示

表4-5顶点赋值测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 赋值失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，0，50 | 赋值失败给出提示 | 8O429$DCJ`S){9U6S}$ITT8  符合理论结果 |
| 用例三 | 测试文件，4，100 | 赋值成功给出提示 | PT_I`KQ~E8GW`QM}[%GJSSV  符合理论结果 |

**示例6：**

对获得第一邻接点功能的测试

测试用例及结果如表4-6所示

表4-6获得第一邻接点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 获取失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，0 | 获取失败给出提示 | BQFTUNW_D]CU36L}OEOTT6O  符合理论结果 |
| 用例三 | 测试文件，4 | 获取成功，2 | `JND3H1DKBWQGDUR2}$~ISB  符合理论结果 |

**示例7：**

对获得下一邻接点功能的测试

测试用例及结果如表4-7所示

表4-7获得下一邻接点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 获取失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，0 | 获取失败给出提示 | _@~X){]JE$GJV`KB[OAGRFV  符合理论结果 |
| 用例三 | 测试文件，7，3 | 获取成功，6 | M)[@F`UM5]ESX$%MH8961_3  符合理论结果 |
| 用例四 | 测试文件，7，6 | 最后一个邻接点 | LAZ32NX4Z@CICP6}CHL@FL8  符合理论结果 |

**示例8：**

对插入顶点功能的测试

测试用例及结果如表4-8所示

表4-8插入顶点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 插入失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，V9,9 | 插入成功给出提示 | ~)T)H%_~0KD1AN[SYENJ{K9~)T)H%_~0KD1AN[SYENJ{K9  符合理论结果 |
| 用例三 | 空间不足 | 插入失败给出提示 |  |

**示例9：**

对删除顶点功能的测试

测试用例及结果如表4-9所示

表4-9删除顶点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 删除失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，20 | 删除失败给出提示 | ~)T)H%_~0KD1AN[SYENJ{K9S)}}J{7)~ZG6U]DJSQASUEI  符合理论结果 |
| 用例三 | 测试文件，5 | 删除成功给出提示 | ~)T)H%_~0KD1AN[SYENJ{K9`AZ%UEL6Q`@05$DPPWDCE1G  符合理论结果 |

**示例10：**

对获得插入弧功能的测试

测试用例及结果如表4-10所示

表4-10插入弧结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 插入失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，1，8，v18，20 | 插入成功给出提示 | ~)T)H%_~0KD1AN[SYENJ{K9T]FP`~T1~7PQ5EY7F1H6[8K  符合理论结果 |
| 用例三 | 空间不足 | 插入失败给出提示 |  |

**示例11：**

对删除弧结点功能的测试

测试用例及结果如表4-11所示

表4-11删除弧结点测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 删除失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件，1，8 | 删除失败给出提示 | ~)T)H%_~0KD1AN[SYENJ{K9`GT}K5QH]CEP}CY6~W[M7XO  符合理论结果 |
| 用例三 | 测试文件，1，2 | 删除成功给出提示 | ~)T)H%_~0KD1AN[SYENJ{K9XWKSE3NS867N8{9~LOQ6D1Q  符合理论结果 |

**示例12：**

对深度优先搜索遍历功能的测试

测试用例及结果如表4-12所示

表4-12深度优先搜索遍历测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 遍历失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件 | 输出遍历结果 | ~)T)H%_~0KD1AN[SYENJ{K93S9B{TYBT4C)]L8GK]$G5VV  符合理论结果 |

**示例13：**

对广度优先搜索遍历功能的测试

测试用例及结果如表4-13所示

表4-13广度优先搜索遍历测试及结果表

|  |  |  |  |
| --- | --- | --- | --- |
| 测试用例 | 程序输入 | 理论结果 | 运行结果 |
| 用例一 | 未创建图 | 遍历失败给出提示 | F`U%YV~MGN[INQ(8@TFQNNY  符合理论结果 |
| 用例二 | 测试文件 | 输出遍历结果 | ~)T)H%_~0KD1AN[SYENJ{K9LF{G84QJ1MW5%@)KM$ZKJ57  符合理论结果 |

经过实验，程序整体测试效果良好，程序基本满足实验要求。

## 4.4 实验小结

本次实验是第一次构建无向网，有了上一次实验创建二叉树的经验，本次无向网的构建虽然较为繁琐，但是也较为轻车熟路地完成了，图的相关结构体虽然繁多，但是本次实验所用的邻接表型存储结构结构并不复杂，且本次要求的函数中没有很多与弧相关的操作或是要求复杂的操作，所以本次实验在构建函数的过程中并没有遇到太多的问题，调试检查程序时也没有遇到无法解决的问题，可以说是相对较为顺利地一次实验，不过在调试检测程序时暴露出来的一些小的细节的问题和bug也说明我在程序编写过程中考虑的不全面和不仔细，在以后的编程实践过程中，我也会更加地认真严谨，更加熟练地掌握编程技巧。

# 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] 严蔚敏等.数据结构题集(C语言版). 清华大学出版社

**指导教师评定意见**

一、对实验报告的评语

|  |
| --- |
|  |

二、对实验报告评分

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 评分项目  (分值) | 程序内容  (36.8分) | 程序规范  (9.2分) | 报告内容  (36.8分) | 报告规范  (9.2分) | 考勤  （8分） | 逾期扣分 | 合 计  (100分) |
| 得分 |  |  |  |  |  |  |  |

# 附录A 基于顺序存储结构线性表实现的源程序

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define SPEC 2

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

typedef int status;

typedef int ElemType;

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

typedef struct SQlist{

ElemType \* elem;

int length;

int listsize;

}SqList;

status IntiaList(SqList \*L);

status DestroyList(SqList \*L);

status ClearList(SqList \*L);

status ListEmpty(SqList L);

int ListLength(SqList L);

status GetElem(SqList L,int i,ElemType \*e);

int LocateElem(SqList L,ElemType e,status (\*Compare) (ElemType x,ElemType y));

status PriorElem(SqList L,ElemType cur,ElemType \*pre\_e);

status NextElem(SqList L,ElemType cur,ElemType \*next\_e);

status ListInsert(SqList \*L,int i,ElemType e);

status ListDelete(SqList \*L,int i,ElemType \*e);

status ListTrabverse(SqList L);

status Compare(ElemType x,ElemType y);

/\*--------------------------------------------\*/

void main(void){

SqList L; int op = 1, found = 1;

FILE \*fp;

while(op)

{

status flag;

int e = 0, i = 0, loc = 0, cur\_e = 0, pre\_e = 0, next\_e = 0;

system("cls");

printf("\n\n");

printf(" Menu for Linear Table On Sequence Structure \n");

printf("-------------------------------------------------\n");

printf(" 1. IntiaList 8. PriorElem\n");

printf(" 2. DestroyList 9. NextElem\n");

printf(" 3. ClearList 10. ListInsert\n");

printf(" 4. ListEmpty 11. ListDelete\n");

printf(" 5. ListLength 12. ListTrabverse\n");

printf(" 6. GetElem 13. SaveData\n");

printf(" 7. LocateElem 14. LoadData\n");

printf(" 0. Exit\n");

printf("-------------------------------------------------\n");

printf(" 请选择你的操作[0~14]:");

scanf("%d",&op);

switch(op)

{

case 1:

if(IntiaList(&L) == OK)

{

printf("线性表创建成功！\n");

found = 0;

}

else

printf("线性表创建失败！\n");

getchar();

getchar();

break;

case 2:

if(found || DestroyList(&L) == ERROR)

printf("线性表不存在！\n");

else

printf("销毁线性表成功！\n");

getchar();

getchar();

break;

case 3:

if(found || ClearList(&L) == ERROR)

printf("线性表不存在！\n");

else

printf("清空线性表成功！\n");

getchar();

getchar();

break;

case 4:

if(found)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

flag = ListEmpty(L);

if(flag == OK)

printf("线性表为空！\n");

else if(flag == ERROR)

printf("线性表不存在！\n");

else

printf("线性表不为空！\n") ;

getchar();

getchar();

break;

case 5:

if(found || L.elem == NULL)

printf("线性表不存在！\n");

else

printf("线性表中元素个数为 %d\n",ListLength(L));

getchar();

getchar();

break;

case 6:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的元素的位置：");

scanf("%d",&i);

flag = GetElem(L, i, &e);

if(flag == ERROR)

printf("输入位置有误!\n");

else

printf("线性表中第%d个元素为%d\n",i ,e);

getchar();

getchar();

break;

case 7:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的元素：");

scanf("%d",&e);

loc = LocateElem(L,e,Compare);

if(loc == ERROR)

printf("线性表中无满足关系的元素!");

else

printf("线性表中与%d满足的关系的元素的位置为%d\n",e,loc);

getchar();

getchar();

break;

case 8:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的元素：");

scanf("%d",&cur\_e);

flag = PriorElem(L,cur\_e,&pre\_e);

if(flag == ERROR)

printf("该元素不在线性表中！\n");

else if(flag == SPEC)

printf("该元素为第一个元素，无前驱！\n");

else

printf("该元素在线性表中的前驱为%d\n",pre\_e);

getchar();

getchar();

break;

case 9:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的元素：");

scanf("%d",&cur\_e);

flag = NextElem(L,cur\_e,&next\_e);

if(flag == ERROR)

printf("该元素不在线性表中！\n");

else if(flag == SPEC)

printf("该元素为最后一个元素，无后继！\n");

else

printf("该元素在线性表中的后继为%d\n",next\_e);

getchar();

getchar();

break;

case 10:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

printf("请输入要插入的位置：");

scanf("%d",&i);

printf("请输入要插入的元素：");

scanf("%d",&e);

flag = ListInsert(&L, i, e);

if(flag == ERROR)

printf("扩充失败！\n");

else if(flag == SPEC)

printf("插入失败，位置输入错误！\n") ;

else

printf("插入成功！\n");

getchar();

getchar();

break;

case 11:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

printf("请输入要删除的元素的位置：");

scanf("%d",&i);

flag = ListDelet(&L, i, &e);

if(flag == ERROR)

printf("删除失败，线性表为空无法删除！\n");

else if(flag == SPEC)

printf("删除失败，位置输入错误！\n") ;

else

printf("删除成功！删除的元素值为%d!\n",e);

getchar();

getchar();

break;

case 12:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

if(!ListTrabverse(L))

printf("线性表是空表！\n");

getchar();

getchar();

break;

case 13:

if(found || L.elem == NULL)

{

printf("线性表不存在！\n");

getchar();

getchar();

break;

}

if((fp = fopen("full.txt","w")) == NULL)

{

printf("文件无法打开!\n");

break;

}

for(i = 0;i < L.length;i++)

fprintf(fp, "%d", \*(L.elem + i));

fclose(fp);

printf("文件保存成功！\n");

getchar();

getchar();

break;

case 14:

if((fp = fopen("full.txt","r")) == NULL)

{

printf("文件无法打开!\n");

break;

}

if(found || L.elem == NULL)

{

flag = IntiaList(&L);

found = 0;

}

else

flag = ClearList(&L);

if(flag)

{

i = 0;

while(fscanf(fp, "%d", (L.elem + i)) == 1)

i++;

L.length = i;

fclose(fp);

printf("文件读取成功！\n");

}

getchar();

getchar();

break;

case 0:

break;

}

}

printf("欢迎下次再使用本系统！\n");

}

status IntiaList(SqList \*L)

{

L->elem = (ElemType \*)malloc(LIST\_INIT\_SIZE \* sizeof (ElemType));

if(!L->elem)

exit(OVERFLOW);

L->length = 0;

L->listsize = LIST\_INIT\_SIZE;

return OK;

}

status DestroyList(SqList \*L)

{

if(L->elem == NULL)

return ERROR;

else

{

free(L->elem);

L->elem = NULL;

L->length = 0;

return OK;

}

}

status ClearList(SqList \*L)

{

if(L->elem == NULL)

return ERROR;

else

{

L->length = 0;

return OK;

}

}

status ListEmpty(SqList L)

{

if(L.elem == NULL)

return ERROR;

else

{

if(L.length == 0)

return OK;

else

return SPEC;

}

}

int ListLength(SqList L)

{

return L.length;

}

status GetElem(SqList L,int i,ElemType \*e)

{

if(i < 1 || i > L.length)

return ERROR;

else

{

\*e = \*(L.elem + i - 1);

return OK;

}

}

int LocateElem(SqList L,int e,status (\*Compare)(ElemType x,ElemType y))

{

int loc = 0;

for(;loc < L.length;loc++)

{

if(Compare(\*(L.elem + loc),e))

return (loc + 1);

}

return ERROR;

}

status PriorElem(SqList L, ElemType cur\_e, ElemType \*pre\_e)

{

int loc = 0;

for(;loc < L.length;loc++)

{

if(\*(L.elem + loc) == cur\_e)

{

if(loc == 0)

return SPEC;

else

{

\*pre\_e = \*(L.elem + loc - 1);

return OK;

}

}

}

return ERROR;

}

status NextElem(SqList L,ElemType cur\_e,ElemType \*next\_e)

{

int loc = 0;

for(;loc < L.length;loc++)

{

if(\*(L.elem + loc) == cur\_e)

{

if(loc == (L.length - 1))

return SPEC;

else

{

\*next\_e = \*(L.elem + loc + 1);

return OK;

}

}

}

return ERROR;

}

status ListInsert(SqList \*L,int i,ElemType e)

{

int loc = L->length;

if(i < 1||i > L->length + 1)

return SPEC;

if((L->length + 1) > L->listsize)

{

ElemType \*newbase;

newbase = (ElemType \*)realloc(L->elem,(++L->listsize) \* sizeof(ElemType));

if(newbase == NULL)

return ERROR;

L->elem = newbase;

while(loc >= i)

{

\*(L->elem + loc) = \*(L->elem + loc - 1);

loc--;

}

\*(L->elem + loc) = e;

L->length++;

return OK;

}

while(loc >= i)

{

\*(L->elem + loc) = \*(L->elem + loc - 1);

loc--;

}

\*(L->elem + loc) = e;

L->length++;

return OK;

}

status ListDelet(SqList \*L,int i,ElemType \*e)

{

int loc = i - 1;

if(ListEmpty(\*L) == OK)

return ERROR;

if(i < 1||i > L->length)

return SPEC;

\*e = \*(L->elem + loc);

while(loc < L->length - 1)

{

\*(L->elem + loc) = \*(L->elem + loc + 1);

loc++;

}

L->length--;

return OK;

}

status ListTrabverse(SqList L)

{

int i;

printf("\n-----------all elements -----------------------\n");

for(i=0;i<L.length;i++)

printf("%d ",L.elem[i]);

printf("\n------------------ end ------------------------\n");

return L.length;

}

status Compare(ElemType x,ElemType y)

{

if(x > y)

return OK;

else

return ERROR;

}

# 附录B 基于顺序存储结构多线性表实现的源程序

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define SPEC 2

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

typedef int status;

typedef int ElemType;

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define MENU\_INIT\_SIZE 20

#define NAME\_MAX\_SIZE 20

typedef struct SQlist{

ElemType \* elem;

int length;

int listsize;

}SqList;

typedef struct MenuName

{

char name[NAME\_MAX\_SIZE];

SqList \*lists;

}MenuName;

typedef struct Menu{

MenuName menuname[MENU\_INIT\_SIZE];

int length;

int menusize;

}Menu;

status FindList(Menu M, char name[NAME\_MAX\_SIZE], int \*e);

status IntiaMenu(Menu \*M);

status IntiaList(SqList \*L);

status DestroyMenu(Menu \*M);

status DestroyList(SqList \*L);

int MenuLength(Menu M);

status MenuTrabverse(Menu M);

status ClearList(SqList \*L);

status ListEmpty(SqList L);

int ListLength(SqList L);

status GetElem(SqList L,int i,ElemType \*e);

int LocateElem(SqList L,ElemType e,status (\*Compare) (ElemType x,ElemType y));

status PriorElem(SqList L,ElemType cur,ElemType \*pre\_e);

status NextElem(SqList L,ElemType cur,ElemType \*next\_e);

status ListInsert(SqList \*L,int i,ElemType e);

status ListDelete(SqList \*L,int i,ElemType \*e);

status ListTrabverse(SqList L);

status ClearMenu(Menu \*M);

status Compare(ElemType x,ElemType y);

/\*--------------------------------------------\*/

void main(void){

Menu M; SqList L; int op = 1, found = 1;

FILE \*fp;

while(op)

{

status flag, flag2;

int e = 0, i = 0, loc = 0, cur\_e = 0, pre\_e = 0, next\_e = 0;

char name[NAME\_MAX\_SIZE];

system("cls");

printf("\n\n");

printf(" Menu for Linear Table On Sequence Structure \n");

printf("-----------------------------------------------------------\n");

printf(" 1. IntiaMenu 11. LocateElem\n");

printf(" 2. IntiaList 12. PriorElem\n");

printf(" 3. DestroyMenu 13. NextElem\n");

printf(" 4. DestroyList 14. ListInsert\n");

printf(" 5. MenuLength 15. ListDelete\n");

printf(" 6. MenuTrabverse 16. ListTrabverse\n");

printf(" 7. ClearList 17. ClearMenu\n");

printf(" 8. ListEmpty 18. LoadData\n");

printf(" 9. ListLength 19. SaveData\n");

printf(" 10. GetElem 0. Exit\n");

printf("-----------------------------------------------------------\n");

printf(" 请选择你的操作[0~19]:");

scanf("%d",&op);

switch(op)

{

case 1:

//IntiaMenu

if(IntiaMenu(&M) == OK)

printf("总线性表创建成功！\n");

else

printf("总线性表创建失败！\n");

getchar();

getchar();

break;

case 2:

//IntiaList

if(IntiaList(&L) == OK)

{

M.menuname[M.length].lists = &L;

printf("请输入创建的线性表的名称：\n");

scanf("%s",M.menuname[M.length].name);

M.length++;

printf("线性表创建成功！\n");

found = 0;

}

else

printf("线性表创建失败！\n");

getchar();

getchar();

break;

case 3:

//DestroyMenu

if(found || DestroyMenu(&M) == ERROR)

printf("总线性表中无线性表！\n");

else

printf("销毁总线性表成功！\n");

getchar();

getchar();

break;

case 4:

//DestroyList

if(found || M.menuname[0].lists== NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入想要销毁的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！\n");

else

{

DestroyList(M.menuname[e].lists);

M.length--;

printf("销毁线性表成功！\n");

}

getchar();

getchar();

break;

case 5:

//MenuLength

if(found || M.menuname[0].lists == NULL)

printf("总线性表中无线性表！\n");

else

printf("线性表中元素个数为 %d\n",MenuLength(M));

getchar();

getchar();

break;

case 6:

//MenuTrabverse

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

if(!MenuTrabverse(M))

printf("线性表是空表！\n");

getchar();

getchar();

break;

case 7:

//ClearList

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入想要清空的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！\n");

else

{

ClearList(M.menuname[e].lists);

printf("清空线性表成功！\n");

}

getchar();

getchar();

break;

case 8:

//ListEmpty

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

flag2 = ListEmpty(L);

if(flag2 == OK)

printf("线性表为空！\n");

else

printf("线性表不为空！\n") ;

}

getchar();

getchar();

break;

case 9:

//ListLength;

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

printf("线性表中元素个数为 %d\n",ListLength(L));

getchar();

getchar();

break;

case 10:

//GetElem

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

printf("请输入要查询的元素的位置：");

scanf("%d",&i);

flag2 = GetElem(L, i, &e);

if(flag2 == ERROR)

printf("输入位置有误!\n");

else

printf("线性表中第%d个元素为%d\n",i ,e);

}

getchar();

getchar();

break;

case 11:

//LocateElem

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

printf("请输入要查询的元素：");

scanf("%d",&e);

loc = LocateElem(L,e,Compare);

if(loc == ERROR)

printf("线性表中无满足关系的元素!");

else

printf("线性表中与%d满足的关系的元素的位置为%d\n",e,loc);

}

getchar();

getchar();

break;

case 12:

//PriorElem

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

printf("请输入要查询的元素：");

scanf("%d",&cur\_e);

flag2 = PriorElem(L,cur\_e,&pre\_e);

if(flag2 == ERROR)

printf("该元素不在线性表中！\n");

else if(flag2 == SPEC)

printf("该元素为第一个元素，无前驱！\n");

else

printf("该元素在线性表中的前驱为%d\n",pre\_e);

}

getchar();

getchar();

break;

case 13:

//NextElem

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

printf("请输入要查询的元素：");

scanf("%d",&cur\_e);

flag2 = NextElem(L,cur\_e,&next\_e);

if(flag2 == ERROR)

printf("该元素不在线性表中！\n");

else if(flag2 == SPEC)

printf("该元素为最后一个元素，无后继！\n");

else

printf("该元素在线性表中的后继为%d\n",next\_e);

}

getchar();

getchar();

break;

case 14:

//ListInsert

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

printf("请输入要插入的位置：");

scanf("%d",&i);

printf("请输入要插入的元素：");

scanf("%d",&e);

flag2 = ListInsert(&L, i, e);

if(flag2 == ERROR)

printf("插入失败，线性表已满无法插入！\n");

else if(flag2 == SPEC)

printf("插入失败，位置输入错误！\n") ;

else

printf("插入成功！\n");

}

getchar();

getchar();

break;

case 15:

//ListDelete

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

printf("请输入要删除的元素的位置：");

scanf("%d",&i);

flag2 = ListDelet(&L, i, &e);

if(flag2 == ERROR)

printf("删除失败，线性表为空无法删除！\n");

else if(flag2 == SPEC)

printf("删除失败，位置输入错误！\n") ;

else

printf("删除成功！删除的元素值为%d!\n",e);

}

getchar();

getchar();

break;

case 16:

//ListTrabverse

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

printf("请输入查询的线性表的名称：\n");

scanf("%s",name);

flag = FindList(M, name, &e);

if(flag == ERROR)

printf("未找到相应名称的线性表！");

else

{

if(!ListTrabverse(L))

printf("线性表是空表！\n");

}

getchar();

getchar();

break;

case 17:

//ClearMenu

if(found || ClearMenu(&M) == ERROR)

printf("总线性表中无线性表！\n");

else

printf("清空线性表成功！\n");

getchar();

getchar();

break;

case 18:

//LoadData

if((fp = fopen("test.txt","r")) == NULL)

{

printf("文件无法打开!\n");

getchar();

getchar();

break;

}

IntiaMenu(&M);

fscanf(fp, "%d", &M.length);

for(i = 0;i < M.length;i++)

{

IntiaList(&L);

M.menuname[i].lists = &L;

}

found = 0;

for(i = 0;i < M.length;i++)

{

fgets(M.menuname[i].name, NAME\_MAX\_SIZE, fp);

strtok(M.menuname[i].name,"\n");

}

for(i = 0;i < M.length;i++)

fscanf(fp, "%d", &M.menuname[i].lists->length);

for(e = 0;e < M.length;e++)

{

for(i = 0;i < M.menuname[e].lists->length;i++)

fscanf(fp, "%d", (M.menuname[e].lists->elem + i));

}

fclose(fp);

printf("文件读取成功！\n");

getchar();

getchar();

break;

case 19:

//SaveData

if(found || M.menuname[0].lists == NULL)

{

printf("总线性表中无线性表！\n");

getchar();

getchar();

break;

}

if((fp = fopen("test.txt","w")) == NULL)

{

printf("文件无法打开!\n");

getchar();

getchar();

break;

}

fprintf(fp, "%d", M.length);

for(i = 0;i < M.length;i++)

fprintf(fp, "%s\n", M.menuname[i].name);

for(i = 0;i < M.length;i++)

fprintf(fp, "%d ", M.menuname[i].lists->length);

for(e = 0;e < M.length;e++)

{

for(i = 0;i < M.menuname[e].lists->length;i++)

fprintf(fp, "%d ", \*(M.menuname[e].lists->elem + i));

}

fclose(fp);

printf("文件保存成功！\n");

getchar();

getchar();

break;

case 0:

break;

}//end of switch

}//end of while

printf("欢迎下次再使用本系统！\n");

}//end of main()

/\*--------page 23 on textbook --------------------\*/

status FindList(Menu M, char name[NAME\_MAX\_SIZE], int \*e)

{

int i = 0;

for(;i < M.length;i++)

{

if(!strcmp(name,M.menuname[i].name))

{

\*e = i;

return OK;

}

}

return ERROR;

}

status IntiaMenu(Menu \*M)

{

int i = 0;

for(;i < MENU\_INIT\_SIZE;i++)

M->menuname[i].lists = NULL;

M->length = 0;

M->menusize = MENU\_INIT\_SIZE;

return OK;

}

status IntiaList(SqList \*L)

{

L->elem = (ElemType \*)malloc(LIST\_INIT\_SIZE \* sizeof (ElemType));

if(!L->elem)

exit(OVERFLOW);

L->length = 0;

L->listsize = LIST\_INIT\_SIZE;

return OK;

}

status DestroyMenu(Menu \*M)

{

int i = M->length - 1;

if(M->menuname[0].lists == NULL)

return ERROR;

else

{

for(;i >= 0;i--)

M->menuname[i].lists = NULL;

M->length = 0;

return OK;

}

}

status DestroyList(SqList \*L)

{

if(L->elem == NULL)

return ERROR;

else

{

free(L->elem);

L->elem = NULL;

L->length = 0;

return OK;

}

}

int MenuLength(Menu M)

{

return M.length;

}

status MenuTrabverse(Menu M)

{

int i;

printf("\n-----------all lists' names -----------------------\n");

for(i=0;i<M.length;i++)

printf("%s ",M.menuname[i].name);

printf("\n------------------ end ------------------------\n");

return M.length;

}

status ClearList(SqList \*L)

{

if(L->elem == NULL)

return ERROR;

else

{

L->length = 0;

return OK;

}

}

status ListEmpty(SqList L)

{

if(L.elem == NULL)

return ERROR;

else

{

if(L.length == 0)

return OK;

else

return SPEC;

}

}

int ListLength(SqList L)

{

return L.length;

}

status GetElem(SqList L,int i,ElemType \*e)

{

if(i < 1 || i > L.length)

return ERROR;

else

{

\*e = \*(L.elem + i - 1);

return OK;

}

}

int LocateElem(SqList L,int e,status (\*Compare)(ElemType x,ElemType y))

{

int loc = 0;

for(;loc < L.length;loc++)

{

if(Compare(\*(L.elem + loc),e))

return (loc + 1);

}

return ERROR;

}

status PriorElem(SqList L, ElemType cur\_e, ElemType \*pre\_e)

{

int loc = 0;

for(;loc < L.length;loc++)

{

if(\*(L.elem + loc) == cur\_e)

{

if(loc == 0)

return SPEC;

else

{

\*pre\_e = \*(L.elem + loc - 1);

return OK;

}

}

}

return ERROR;

}

status NextElem(SqList L,ElemType cur\_e,ElemType \*next\_e)

{

int loc = 0;

for(;loc < L.length;loc++)

{

if(\*(L.elem + loc) == cur\_e)

{

if(loc == (L.length - 1))

return SPEC;

else

{

\*next\_e = \*(L.elem + loc + 1);

return OK;

}

}

}

return ERROR;

}

status ListInsert(SqList \*L,int i,ElemType e)

{

int loc = L->length;

if(L->length == L->listsize)

return ERROR;

if(i < 1||i > L->length + 1)

return SPEC;

while(loc >= i)

{

\*(L->elem + loc) = \*(L->elem + loc - 1);

loc--;

}

\*(L->elem + loc) = e;

L->length++;

return OK;

}

status ListDelet(SqList \*L,int i,ElemType \*e)

{

int loc = i - 1;

if(ListEmpty(\*L) == OK)

return ERROR;

if(i < 1||i > L->length)

return SPEC;

\*e = \*(L->elem + loc);

while(loc < L->length - 1)

{

\*(L->elem + loc) = \*(L->elem + loc + 1);

loc++;

}

L->length--;

return OK;

}

status ListTrabverse(SqList L)

{

int i;

printf("\n-----------all elements -----------------------\n");

for(i=0;i<L.length;i++)

printf("%d ",L.elem[i]);

printf("\n------------------ end ------------------------\n");

return L.length;

}

status ClearMenu(Menu \*M)

{

if(M->menuname[0].lists == NULL)

return ERROR;

else

{

M->length = 0;

return OK;

}

}

status Compare(ElemType x,ElemType y)

{

if(x == y)

return OK;

else

return ERROR;

}

**附录C 基于链式存储结构线性表实现的源程序**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define SPEC 2

typedef int status;

typedef int ElemType;

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define NAME\_MAX\_SIZE 20

typedef struct Lnode{

ElemType data;

struct Lnode \*next;

}Lnode, \*Linklist;

typedef struct Lname{

char name[NAME\_MAX\_SIZE];

Lnode \*lists;

}Lname;

typedef struct Slist{

Lname listname;

struct Slist \*next;

}Slist, \*LinkSlist;

#define LENG sizeof(struct Lnode)

#define SLENG sizeof(struct Slist)

status IntiaList(Lnode \*\*head);

status DestroyList(Lnode \*\*head);

status ClearList(Lnode \*head);

status ListEmpty(Lnode \*head);

int ListLength(Lnode \*head);

status GetElem(Lnode \*head,int i,ElemType \*e);

int LocateElem(Lnode \*head,ElemType e,status (\*Compare) (ElemType x,ElemType y));

status PriorElem(Lnode \*head,ElemType cur\_e,ElemType \*pre\_e);

status NextElem(Lnode \*head,ElemType cur\_e,ElemType \*next\_e);

status ListInsert(Lnode \*head,int i,ElemType e);

status ListDelete(Lnode \*head,int i,ElemType \*e);

status ListTrabverse(Lnode \*head);

status Compare(ElemType x,ElemType y);

status IntiaSList(Slist \*\*shead);

status DestroySList(Slist \*\*shead);

status SListInsert(Slist \*shead, int i, char name[NAME\_MAX\_SIZE]);

status SListDelete(Slist \*shead, int i, char name[NAME\_MAX\_SIZE]);

void main(void){

Linklist head = NULL;

LinkSlist shead = NULL;

int op=1;

FILE \*fp;

while(op)

{

int e = 0, i = 0, flag = 0, cur\_e = 0, pre\_e = 0, next\_e = 0;

char name[NAME\_MAX\_SIZE];

system("cls"); printf("\n\n");

printf(" Menu for Chain Table On Sequence Structure \n");

printf("-------------------------------------------------\n");

printf(" 1. IntiaList 7. LocateElem\n");

printf(" 2. DestroyList 8. PriorElem\n");

printf(" 3. ClearList 9. NextElem \n");

printf(" 4. ListEmpty 10. ListInsert\n");

printf(" 5. ListLength 11. ListDelete\n");

printf(" 6. GetElem 12. ListTrabverse\n");

printf("-------------------------------------------------\n");

printf(" 13. DataLoad 16. DestroySList\n");

printf(" 14. DataSave 17. SListInsert\n");

printf(" 15. IntiaSList 18. SListDelete\n");

printf(" 0. Exit\n");

printf("-------------------------------------------------\n");

printf(" 请选择你的操作[0~18]:");

scanf("%d",&op);

if(op > 0 && op < 15)

{

if(shead != NULL)

{

head = NULL;

if(op == 1)

{

printf("表已创建！\n");

getchar();

getchar();

continue;

}

char name[NAME\_MAX\_SIZE];

printf("请输入要操作的表的名称！\n");

scanf("%s",name);

Slist \*loc = shead->next;

while(loc != NULL)

{

if(!strcmp(loc->listname.name,name))

break;

loc = loc->next;

}

if(loc == NULL)

{

printf("所输入表不存在！\n");

getchar();

getchar();

continue;

}

head = loc->listname.lists;

}

}

switch(op){

case 1:

//IntiaList

if(IntiaList(&head)==OK)

printf("线性表创建成功！\n");

else

printf("线性表创建失败！\n");

getchar();

getchar();

break;

case 2:

//DestroyList

if(head == NULL)

printf("线性表未创建！\n");

else

{

DestroyList(&head);

printf("线性表销毁成功！\n");

}

getchar();

getchar();

break;

case 3:

//ClearList

if(head == NULL)

printf("线性表未创建！\n");

else

{

ClearList(head);

printf("线性表清除成功！\n");

}

getchar();

getchar();

break;

case 4:

//ListEmpty

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

if(ListEmpty(head) == OK)

printf("线性表为空！\n");

else

printf("线性表不为空！\n");

getchar();

getchar();

break;

case 5:

//ListLength

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

printf("线性表长度为%d!\n",ListLength(head));

getchar();

getchar();

break;

case 6:

//GetElem

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要查询的元素的位置：\n");

scanf("%d",&i);

if(GetElem(head, i, &e) == ERROR)

printf("线性表中无对应元素!\n");

else

printf("线性表中%d位置的元素为%d\n",i,e);

getchar();

getchar();

break;

case 7:

//LocateElem

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要查询的元素：\n");

scanf("%d",&e);

i = LocateElem(head, e, Compare);

if(i == ERROR)

printf("线性表中无对应元素!\n");

else

printf("线性表中满足所给关系的元素的位置为%d\n",i);

getchar();

getchar();

break;

case 8:

//PriorElem

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要查询的元素：\n");

scanf("%d",&cur\_e);

flag = PriorElem(head, cur\_e, &pre\_e);

if(flag == ERROR)

printf("线性表中无对应元素!\n");

else if(flag == SPEC)

printf("此元素为第一个元素，无前驱！\n");

else

printf("元素%d的前驱为%d!\n",cur\_e,pre\_e);

getchar();

getchar();

break;

case 9:

//NextElem

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要查询的元素：\n");

scanf("%d",&cur\_e);

flag = NextElem(head, cur\_e, &next\_e);

if(flag == ERROR)

printf("线性表中无对应元素!\n");

else if(flag == SPEC)

printf("此元素为最后一个元素，无后继！\n");

else

printf("元素%d的后继为%d!\n",cur\_e,next\_e);

getchar();

getchar();

break;

case 10:

//ListInsert

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要插入的元素的位置：\n");

scanf("%d",&i);

printf("请输入想要插入的元素：\n");

scanf("%d",&e);

flag = ListInsert(head, i, e);

if(flag == ERROR)

printf("插入失败! 输入位置错误!\n");

else if(flag == OVERFLOW)

printf("插入失败! 空间溢出!\n");

else

printf("插入成功!\n");

getchar();

getchar();

break;

case 11:

//ListDelet

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要删除的元素的位置：\n");

scanf("%d",&i);

flag = ListDelete(head, i, &e);

if(flag == ERROR)

printf("删除失败! 输入位置错误!\n");

else

printf("删除成功! 所删除的元素为%d!\n",e);

getchar();

getchar();

break;

case 12:

//ListTrabverse

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

if(!ListTrabverse(head)) printf("线性表是空表！\n");

getchar();

getchar();

break;

case 0:

break;

case 13:

//DataLoad

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

if((fp = fopen("test.txt","r")) == NULL)

{

printf("文件无法打开!\n");

break;

}

for(i = 1; fscanf(fp, "%d", &e) == 1; i++)

ListInsert(head, i, e);

fclose(fp);

printf("文件读取成功！\n");

getchar();

getchar();

break;

case 14:

//DataSave

if(head == NULL)

{

printf("线性表未创建！\n");

getchar();

getchar();

break;

}

if((fp = fopen("stest.txt","w")) == NULL)

{

printf("文件无法打开!\n");

break;

}

Lnode \*n =head->next;

while(n != NULL)

{

fprintf(fp, "%d ", n->data);

n = n->next;

}

fclose(fp);

printf("文件保存成功！\n");

getchar();

getchar();

break;

case 15:

//IntiaSList

if(IntiaSList(&shead)==OK)

printf("总线性表创建成功！\n");

else

printf("总线性表创建失败！\n");

getchar();

getchar();

break;

case 16:

//DestroySList

if(shead == NULL)

printf("总线性表未创建！\n");

else

{

DestroySList(&shead);

printf("总线性表销毁成功！\n");

}

getchar();

getchar();

break;

case 17:

//SListInsert

if(shead == NULL)

{

printf("总线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要插入的表的位置：\n");

scanf("%d",&i);

printf("请输入想要插入的表的名称：\n");

scanf("%s",name);

flag = SListInsert(shead, i, name);

if(flag == ERROR)

printf("插入失败! 输入位置错误!\n");

else if(flag == OVERFLOW)

printf("插入失败! 空间溢出!\n");

else

printf("插入成功!\n");

getchar();

getchar();

break;

case 18:

//SListDelete

if(shead == NULL)

{

printf("总线性表未创建！\n");

getchar();

getchar();

break;

}

printf("请输入想要删除的表的位置：\n");

scanf("%d",&i);

flag = SListDelete(shead, i, name);

if(flag == ERROR)

printf("删除失败! 输入位置错误!\n");

else

printf("删除成功! 所删除的元素为%s!\n",name);

getchar();

getchar();

break;

}

}

printf("欢迎下次再使用本系统！\n");

}

status IntiaList(Lnode \*\*head){

Lnode \*L;

L = (Lnode \*)malloc(LENG);

if(!L)

return OVERFLOW;

L->next = NULL;

\*head = L;

return OK;

}

status IntiaSList(Slist \*\*shead){

Slist \*S;

S = (Slist \*)malloc(SLENG);

if(!S)

return OVERFLOW;

S->next = NULL;

\*shead = S;

return OK;

}

status DestroyList(Lnode \*\*head){

Lnode \*p , \*n;

p = \*head;

n = p->next;

while(n != NULL)

{

free(p);

p = n;

n = n->next;

}

free(p);

\*head = NULL;

return OK;

}

status DestroySList(Slist \*\*shead){

Slist \*s, \*t;

s = \*shead;

t = s->next;

if(t != NULL)

{

while(t != NULL)

{

DestroyList(&(s->listname.lists));

free(s);

s = t;

t = t->next;

}

free(s);

}

else

free(s);

\*shead = NULL;

return OK;

}

status ClearList(Lnode \*head){

Lnode \*p, \*n;

p = head;

n = p->next;

if(n == NULL)

return OK;

else

{

p = n;

n = n->next;

while(n != NULL)

{

free(p);

p = n;

n = n->next;

}

free(p);

}

head->next = NULL;

}

status ListEmpty(Lnode \*head){

Lnode \*p = head->next;

if(p != NULL)

return ERROR;

else

return OK;

}

int ListLength(Lnode \*head)

{

Lnode \*p = head->next;

int length = 0;

while(p != NULL)

{

length++;

p = p->next;

}

return length;

}

status GetElem(Lnode \*head, int i, ElemType \*e){

int length = ListLength(head);

if(i < 1 || i > length)

return ERROR;

Lnode \*n;

n = head;

while(i--)

n = n->next;

\*e = n->data;

return OK;

}

int LocateElem(Lnode \*head,int e,status (\*Compare)(ElemType x,ElemType y)){

Lnode \*n = head->next;

int loc = 1;

while(n != NULL)

{

if(Compare(n->data, e))

return loc;

loc++;

n = n->next;

}

return ERROR;

}

status PriorElem(Lnode \*head, ElemType cur\_e, ElemType \*pre\_e){

Lnode \*n = head->next, \*p = head;

if(n == NULL)

return ERROR;

else

{

if(n->data == cur\_e)

return SPEC;

else

{

while(n->next != NULL)

{

p = n;

n = n->next;

if(n->data == cur\_e)

{

\*pre\_e = p->data;

return OK;

}

}

return ERROR;

}

}

}

status NextElem(Lnode \*head, ElemType cur\_e, ElemType \*next\_e){

Lnode \*n = head->next;

if(n == NULL)

return ERROR;

else

{

while(n->next != NULL)

{

if(n->data == cur\_e)

{

\*next\_e = n->next->data;

return OK;

}

n = n->next;

}

if(n->data == cur\_e)

return SPEC;

return ERROR;

}

}

status ListInsert(Lnode \*head, int i, ElemType e){

if(i < 1)

return ERROR;

Lnode \*n = head, \*newnode;

while(--i)

{

n = n->next;

if(n == NULL)

return ERROR;

}

newnode = (Lnode \*)malloc(LENG);

if(!newnode)

return OVERFLOW;

newnode->next = n->next;

n->next = newnode;

newnode->data = e;

return OK;

}

status SListInsert(Slist \*shead, int i, char name[NAME\_MAX\_SIZE]){

if(i < 1)

return ERROR;

Slist \*n = shead, \*newnode;

while(--i)

{

n = n->next;

if(n == NULL)

return ERROR;

}

newnode = (Slist \*)malloc(SLENG);

if(!newnode)

return OVERFLOW;

newnode->next = n->next;

n->next = newnode;

strcpy(newnode->listname.name, name);

IntiaList(&(newnode->listname.lists));

return OK;

}

status ListDelete(Lnode \*head, int i, ElemType \*e){

if(i < 1)

return ERROR;

Lnode \*n = head, \*denode;

while(--i)

{

n = n->next;

if(n->next == NULL)

return ERROR;

}

denode = n->next;

\*e = denode->data;

n->next = n->next->next;

free(denode);

return OK;

}

status SListDelete(Slist \*shead, int i, char name[NAME\_MAX\_SIZE]){

if(i < 1)

return ERROR;

Slist \*n = shead, \*denode;

while(--i)

{

n = n->next;

if(n == NULL)

return ERROR;

}

denode = n->next;

strcpy(name,denode->listname.name);

n->next = n->next->next;

free(denode);

return OK;

}

status ListTrabverse(Lnode \*head){

Lnode \*n = head->next;

if(n == NULL)

return ERROR;

printf("\n-----------all elements -----------------------\n");

while(n != NULL)

{

printf("%d ",n->data);

n = n->next;

}

printf("\n------------------ end ------------------------\n");

return OK;

}

status Compare(ElemType x,ElemType y)

{

if(x > y)

return OK;

else

return ERROR;

}

**附录D 基于二叉链表二叉树实现的源程序**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define SPEC -2

#define NAME\_MAX\_SIZE 20

#define TEL\_MAX\_SIZE 15

#define TREE\_MAX\_SIZE 200

#define TREE\_NAME\_SIZE 20

#define MENU\_INIT\_SIZE 50

typedef int status;

typedef char ElemType1;

typedef char ElemType2;

typedef struct BiTNode{

ElemType1 name[NAME\_MAX\_SIZE];

ElemType2 tel[TEL\_MAX\_SIZE];

struct BiTNode \*lchild;

struct BiTNode \*rchild;

}BiTNode;

typedef struct MenuName

{

char name[TREE\_NAME\_SIZE];

BiTNode \*bitree;

}MenuName;

typedef struct Menu{

MenuName \*menuname;

int length;

int menusize;

}Menu;

int flag = 0;

static int count = 0;

static char menu\_name[TREE\_NAME\_SIZE];

#define LENG sizeof(BiTNode)

status InitBiTree(BiTNode \*\*T);

status DestroyBiTree(BiTNode \*\*T);

status CreateBiTree(BiTNode \*\*T, ElemType1 definition[2][TREE\_MAX\_SIZE][NAME\_MAX\_SIZE]);

status ClearBiTree(BiTNode \*\*T);

status BiTreeEmpty(BiTNode \*T);

int BiTreeDepth(BiTNode \*T);

BiTNode \*Root(BiTNode \*T);

ElemType2 \*Value(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

status Assign(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE], ElemType2 value[TEL\_MAX\_SIZE]);

BiTNode \*Parent(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*LeftChild(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*RightChild(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*LeftSibling(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

BiTNode \*RightSibling(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

status InsertChild(BiTNode \*T, BiTNode \*p, int LR, BiTNode \*c);

status DeleteChild(BiTNode \*T, BiTNode \*p, int LR);

status PreOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

status InOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

status PostOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

status LevelOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T));

void Visit(BiTNode \*T);

BiTNode \*Find(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]);

status Output(BiTNode \*T, FILE \*fp);

status MenuInsert(Menu \*M, BiTNode \*T, char name[TREE\_NAME\_SIZE]);

status NewBiTree(BiTNode \*\*T);

int FindBiTree(Menu M, char name[TREE\_NAME\_SIZE]);

BiTNode \*ChooseBiTree(Menu M, char name[TREE\_NAME\_SIZE]);

status MenuDelete(Menu \*M, char name[TREE\_NAME\_SIZE]);

void main(void){

int i = 0;

Menu M;

BiTNode \*T;

int op = 1, length;

FILE \*fp;

ElemType1 definition[2][TREE\_MAX\_SIZE][NAME\_MAX\_SIZE];

M.menuname = (MenuName \*)malloc(MENU\_INIT\_SIZE \* sizeof(MenuName));

M.length = 0;

M.menusize = MENU\_INIT\_SIZE;

while(op)

{

int i = 0, p = 0;

for(i = 0;i < TREE\_MAX\_SIZE;i++)

{

strcpy(definition[0][i], "#");

strcpy(definition[1][i], "#");

}

i = 0;

ElemType1 e[NAME\_MAX\_SIZE];

ElemType2 t[TEL\_MAX\_SIZE];

BiTNode \*N = NULL;

char tree\_name[TREE\_NAME\_SIZE];

system("cls"); printf("\n\n");

printf(" Menu for Binary Tree On Sequence Structure \n");

printf("-------------------------------------------------\n");

printf(" 1. InitBiTree 11. LeftChild\n");

printf(" 2. DestroyBiTree 12. RightChild\n");

printf(" 3. CreateBiTree 13. LeftSibling \n");

printf(" 4. ClearBiTree 14. RightSibling\n");

printf(" 5. BiTreeEmpty 15. InsertChild\n");

printf(" 6. BiTreeDepth 16. DeleteChild\n");

printf(" 7. Root 17. PreOrderTraverse\n");

printf(" 8. Value 18. InOrderTraverse\n");

printf(" 9. Assign 19. PostOrderTraverse\n");

printf(" 10. Parent 20. LevelOrderTraverse\n");

printf("-------------------------------------------------\n");

printf(" 21. DataSave 24. MenuDelete\n");

printf(" 22. DataLoad 25. NewTree\n");

printf(" 23. MenuInsert 26. ChooseTree\n");

printf(" 0. Exit\n");

printf("-------------------------------------------------\n");

printf(" 请选择你的操作[0~26]:");

scanf("%d",&op);

switch(op){

case 1:

//InitBiTree

InitBiTree(&T);

printf("二叉树创建成功！\n");

getchar();

getchar();

break;

case 2:

//DestroyBiTree

if(DestroyBiTree(&T) == ERROR)

printf("二叉树未创建！\n");

else

printf("二叉树销毁成功！\n");

M.menuname[FindBiTree(M, menu\_name)].bitree = NULL;

getchar();

getchar();

break;

case 3:

//CreateBiTree

printf("请输入二叉树长度：\n");

scanf("%d", &length);

for(i = 0;i < length;i++)

{

printf("请输入结点的名称：\n");

scanf("%s", definition[0][i]);

if(strcmp(definition[0][i], "#"))

{

printf("请输入结点的值：\n");

scanf("%s", definition[1][i]);

}

}

CreateBiTree(&T, definition);

printf("二叉树创建成功！\n");

count = 0;

getchar();

getchar();

break;

case 4:

//ClearBiTree

p = ClearBiTree(&T);

if(p == ERROR)

printf("二叉树未创建！\n");

else if(p == SPEC)

printf("二叉树已为空！\n");

else

printf("二叉树清空成功！\n");

M.menuname[FindBiTree(M, menu\_name)].bitree = NULL;

getchar();

getchar();

break;

case 5:

//BiTreeEmpty

i = BiTreeEmpty(T);

if(i == SPEC)

printf("二叉树未创建！\n");

else if(i == FALSE)

printf("二叉树不为空！\n");

else if(i == TRUE)

printf("二叉树为空！\n");

getchar();

getchar();

break;

case 6:

//BiTreeDepth

i = BiTreeDepth(T);

if(i == SPEC)

printf("二叉树未创建！\n");

else if(i == ERROR)

printf("二叉树为空！\n");

else

printf("二叉树深度为%d！\n", i);

getchar();

getchar();

break;

case 7:

//Root

printf("根节点为：\n");

Visit(Root(T));

getchar();

getchar();

break;

case 8:

//Value

if(flag == 0)

{

printf("二叉树未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的结点的名称：\n");

scanf("%s", e);

if(Find(T, e) == NULL)

{

printf("未找到相应结点！\n");

getchar();

getchar();

break;

}

printf("查询的结点的值为%s\n", Value(T, e));

getchar();

getchar();

break;

case 9:

//Assign

if(flag == 0)

{

printf("二叉树未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要赋值的结点的名称：\n");

scanf("%s", e);

printf("请输入要赋予的值：\n");

scanf("%s", t);

if(Find(T, e) == NULL)

{

printf("未找到相应结点！\n");

getchar();

getchar();

break;

}

if(Assign(T, e, t) == ERROR)

printf("未找到相应结点！\n");

else

printf("赋值成功！\n");

getchar();

getchar();

break;

case 10:

//Parent

printf("请输入要查询的结点的名称：\n");

scanf("%s", e);

N = Parent(T, e);

if(N == NULL)

printf("未找到相应的结点或表未创建！\n");

else

printf("%s的双亲结点是%s", e, N->name);

getchar();

getchar();

break;

case 11:

//LeftChild

printf("请输入要查询的结点的名称：\n");

scanf("%s", e);

N = LeftChild(T, e);

if(N == NULL)

printf("未找到相应的结点或表未创建！\n");

else

printf("%s的左孩子结点是%s", e, N->name);

getchar();

getchar();

break;

case 12:

//RightChild

printf("请输入要查询的结点的名称：\n");

scanf("%s", e);

N = RightChild(T, e);

if(N == NULL)

printf("未找到相应的结点或表未创建！\n");

else

printf("%s的右孩子结点是%s", e, N->name);

getchar();

getchar();

break;

case 13:

//LeftSibling

printf("请输入要查询的结点的名称：\n");

scanf("%s", e);

N = LeftSibling(T, e);

if(N == NULL)

printf("未找到相应的结点或表未创建！\n");

else

printf("%s的左兄弟结点是%s", e, N->name);

getchar();

getchar();

break;

case 14:

//RightSibling

printf("请输入要查询的结点的名称：\n");

scanf("%s", e);

N = RightSibling(T, e);

if(N == NULL)

printf("未找到相应的结点或表未创建！\n");

else

printf("%s的右兄弟结点是%s", e, N->name);

getchar();

getchar();

break;

case 15:

//InsertChild

if(flag == 0)

{

printf("二叉树未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要插入的子树的名称：\n");

scanf("%s", e);

printf("请选择插入左子树或者右子树（0为左子树，1为右子树）：\n");

scanf("%d", &p);

if(Find(T, e) == NULL)

{

printf("未找到相应结点！\n");

getchar();

getchar();

break;

}

printf("创建要插入的二叉树\n");

printf("请输入要插入的二叉树长度：\n");

scanf("%d", &length);

for(i = 0;i < length;i++)

{

printf("请输入结点的名称：\n");

scanf("%s", definition[0][i]);

if(strcmp(definition[0][i], "#"))

{

printf("请输入结点的值：\n");

scanf("%s", definition[1][i]);

}

}

CreateBiTree(&N, definition);

count = 0;

if(InsertChild(T, Find(T, e), p, N) == SPEC)

printf("二叉树未创建！\n");

else

printf("插入子树成功！\n");

getchar();

getchar();

break;

case 16:

//DeleteChild

if(flag == 0)

{

printf("二叉树未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要删除的子树的名称：\n");

scanf("%s", e);

printf("请选择删除左子树或者右子树（0为左子树，1为右子树）：\n");

scanf("%d", &i);

if(Find(T, e) == NULL)

{

printf("未找到相应结点！\n");

getchar();

getchar();

break;

}

if(DeleteChild(T, Find(T, e), i) == SPEC)

printf("删除失败！\n");

else

printf("删除成功！\n");

getchar();

getchar();

break;

case 17:

//PreOrderTraverse

if(PreOrderTraverse(T, Visit) == ERROR)

printf("二叉树为空或未创建！\n");

else

printf("以上为先序遍历！\n");

getchar();

getchar();

break;

case 18:

//InOrderTraverse

if(InOrderTraverse(T, Visit) == ERROR)

printf("二叉树为空或未创建！\n");

else

printf("以上为中序遍历！\n");

getchar();

getchar();

break;

case 19:

//PostOrderTraverse

if(PostOrderTraverse(T, Visit) == ERROR)

printf("二叉树为空或未创建！\n");

else

printf("以上为后序遍历！\n");

getchar();

getchar();

break;

case 20:

//LevelOrderTraverse

if(LevelOrderTraverse(T, Visit) == ERROR)

printf("二叉树为空或未创建！\n");

else

printf("以上为层序遍历！\n");

getchar();

getchar();

break;

case 21:

//DataSave

if((fp = fopen("test.txt","w")) == NULL)

{

printf("文件无法打开!\n");

break;

}

p = Output(T, fp);

if(p == SPEC)

printf("二叉树未创建！\n");

else if(p == ERROR)

printf("二叉树为空！\n");

else

printf("二叉树保存成功！\n");

fclose(fp);

getchar();

getchar();

break;

case 22:

//DataLoad

if((fp = fopen("test.txt","r")) == NULL)

{

printf("文件无法打开!\n");

break;

}

while(1)

{

if(fscanf(fp, "%s", definition[0][i]) == EOF)

break;

if(strcmp(definition[0][i], "#"))

fscanf(fp, "%s", definition[1][i]);

i++;

}

CreateBiTree(&T, definition);

printf("二叉树读取成功！\n");

count = 0;

fclose(fp);

getchar();

getchar();

break;

case 23:

//MenuInsert

if(flag == 0)

{

printf("二叉树未创建！\n");

getchar();

getchar();

break;

}

printf("请输入二叉树的名称！\n");

scanf("%s", tree\_name);

MenuInsert(&M, T, tree\_name);

printf("二叉树插入成功！\n");

strcpy(menu\_name, tree\_name);

getchar();

getchar();

break;

case 24:

//MenuDelete

printf("请输入要删除的二叉树的名称\n");

scanf("%s", tree\_name);

if(MenuDelete(&M, tree\_name) == ERROR)

printf("未找到相应名称的二叉树！\n");

else

printf("二叉树删除成功！\n");

getchar();

getchar();

break;

case 25:

//NewTree

NewBiTree(&T);

printf("已创建新的二叉树！\n");

getchar();

getchar();

break;

case 26:

//ChooseBiTree

printf("请输入要操作的二叉树的名称！\n");

scanf("%s", tree\_name);

N = ChooseBiTree(M, tree\_name);

if(N == NULL && FindBiTree(M, tree\_name) == -1)

printf("未找到相应名称的二叉树！\n");

else

{

T = N;

strcpy(menu\_name, tree\_name);

printf("已成功找到二叉树，请进行操作！\n");

}

getchar();

getchar();

break;

}

}

printf("欢迎下次再使用本系统！\n");

}

status InitBiTree(BiTNode \*\*T){

\*T = NULL;

flag = 1;

return OK;

/\* if(\*T != NULL)

retrun ERROR;

BiTNode \*N;

N = (BiTNode \*)malloc(LENG);

if(!N)

return OVERFLOW;

N->lchild = NULL;

N->rchild = NULL;

N->name = NULL;

N->tel = NULL;

\*T = N;

return OK;

\*/

}

status DestroyBiTree(BiTNode \*\*T){

if(flag == 0 )

return ERROR;

if(\*T == NULL)

{

flag = 0;

return OK;

}

BiTNode \*N = \*T;

if(N->lchild != NULL)

DestroyBiTree(&N->lchild);

if(N->rchild != NULL)

DestroyBiTree(&N->rchild);

free(N);

N = NULL;

\*T = N;

flag = 0;

return OK;

}

status CreateBiTree(BiTNode \*\*T, ElemType1 definition[2][TREE\_MAX\_SIZE][NAME\_MAX\_SIZE]){

BiTNode \*N = \*T;

if (!strcmp(definition[0][count], "#"))

{

N = NULL;

count++;

}

else

{

if (!(N = (BiTNode \*) malloc(sizeof(BiTNode))))

exit(OVERFLOW);

strcpy(N->name, definition[0][count]);

strcpy(N->tel, definition[1][count]);

count++;

CreateBiTree(&N->lchild, definition);

CreateBiTree(&N->rchild, definition);

}

\*T = N;

flag = 1;

return OK;

}

status ClearBiTree(BiTNode \*\*T){

if(\*T == NULL)

return SPEC;

if(flag == 0)

return ERROR;

BiTNode \*N = \*T;

if(N->lchild != NULL)

ClearBiTree(&N->lchild);

if(N->rchild != NULL)

ClearBiTree(&N->rchild);

free(N);

N = NULL;

\*T = N;

return OK;

}

status BiTreeEmpty(BiTNode \*T){

if(flag == 0)

return SPEC;

else if(T == NULL)

return TRUE;

else

return FALSE;

}

int BiTreeDepth(BiTNode \*T){

if(flag == 0)

return SPEC;

if(T == NULL)

return 0;

int depth, left\_depth, right\_depth;

left\_depth = BiTreeDepth(T->lchild);

right\_depth = BiTreeDepth(T->rchild);

if(left\_depth > right\_depth)

depth = 1 + left\_depth;

else

depth = 1 + right\_depth;

return depth;

}

BiTNode \*Root(BiTNode \*T){

if(flag == 0)

return NULL;

return T;

}

ElemType2 \*Value(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]){

ElemType2 \*N = NULL;

ElemType2 \*P = NULL;

if(T == NULL)

return NULL;

if(!strcmp(T->name, e))

N = T->tel;

P = Value(T->lchild, e);

if(P)

N = P;

P = Value(T->rchild, e);

if(P)

N = P;

return N;

}

status Assign(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE], ElemType2 value[TEL\_MAX\_SIZE]){

if(T == NULL)

return ERROR;

else

{

if(!strcmp(T->name, e))

strcpy(T->tel, value);

Assign(T->lchild, e, value);

Assign(T->rchild, e, value);

}

return OK;

}

BiTNode \*Parent(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]){

BiTNode \*N = NULL, \*P = NULL;

if(T == NULL)

return NULL;

if(T->lchild != NULL)

{

if(!strcmp(T->lchild->name, e))

N = T;

}

if(T->rchild != NULL)

{

if(!strcmp(T->rchild->name, e))

N = T;

}

P = Parent(T->lchild, e);

if(P)

N = P;

P = Parent(T->rchild, e);

if(P)

N = P;

return N;

}

BiTNode \*LeftChild(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]){

BiTNode \*N = NULL, \*P = NULL;

if(T == NULL)

return NULL;

if(!strcmp(T->name, e))

N = T->lchild;

P = LeftChild(T->lchild, e);

if(P)

N = P;

P = LeftChild(T->rchild, e);

if(P)

N = P;

return N;

}

BiTNode \*RightChild(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]){

BiTNode \*N = NULL, \*P = NULL;

if(T == NULL)

return NULL;

if(!strcmp(T->name, e))

N = T->rchild;

P = RightChild(T->lchild, e);

if(P)

N = P;

P = RightChild(T->rchild, e);

if(P)

N = P;

return N;

}

BiTNode \*LeftSibling(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]){

BiTNode \*N = NULL, \*P = NULL;

if(T == NULL)

return NULL;

if(T->lchild != NULL)

{

if(!strcmp(T->lchild->name, e))

N = NULL;

}

if(T->rchild != NULL)

{

if(!strcmp(T->rchild->name, e))

N = T->lchild;

}

P = LeftSibling(T->lchild, e);

if(P)

N = P;

P = LeftSibling(T->rchild, e);

if(P)

N = P;

return N;

}

BiTNode \*RightSibling(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE]){

BiTNode \*N = NULL, \*P = NULL;

if(T == NULL)

return NULL;

if(T->rchild != NULL)

{

if(!strcmp(T->rchild->name, e))

N = NULL;

}

if(T->lchild != NULL)

{

if(!strcmp(T->lchild->name, e))

N = T->rchild;

}

P = RightSibling(T->lchild, e);

if(P)

N = P;

P = RightSibling(T->rchild, e);

if(P)

N = P;

return N;

}

status InsertChild(BiTNode \*T, BiTNode \*p, int LR, BiTNode \*c){

if(LR == 0)

{

c->rchild = p->lchild;

p->lchild = c;

}

else if(LR == 1)

{

c->rchild = p->rchild;

p->rchild = c;

}

return OK;

}

status DeleteChild(BiTNode \*T, BiTNode \*p, int LR){

if(LR == 0)

{

if(p->lchild != NULL)

ClearBiTree(&(p->lchild));

else

{

printf("左子树为空！\n");

return SPEC;

}

}

if(LR == 1)

{

if(p->rchild != NULL)

ClearBiTree(&(p->rchild));

else

{

printf("右子树为空！\n");

return SPEC;

}

}

return OK;

}

status PreOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T)){

if(T == NULL)

return ERROR;

else

{

Visit(T);

PreOrderTraverse(T->lchild, Visit);

PreOrderTraverse(T->rchild, Visit);

}

return OK;

}

status InOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T)){

BiTNode \*st[TREE\_MAX\_SIZE + 1];

int top = 0;

if(T == NULL)

return ERROR;

while(T || top)

{

while(T)

{

if(top == TREE\_MAX\_SIZE)

exit(OVERFLOW);

st[++top] = T;

T = T->lchild;

}

if(top)

{

T = st[top--];

Visit(T);

T = T->rchild;

}

}

return OK;

}

status PostOrderTraverse(BiTNode \*T,void (\*Visit)(BiTNode \*T)){

if(T == NULL)

return ERROR;

else

{

PostOrderTraverse(T->lchild, Visit);

PostOrderTraverse(T->rchild, Visit);

Visit(T);

}

return OK;

}

status LevelOrderTraverse(BiTNode \*T, void (\*Visit)(BiTNode \*T)){

BiTNode \*st[TREE\_MAX\_SIZE + 1];

int top = 0, tail = 1;

if(T == NULL)

return ERROR;

st[++top] = T;

while(top <= tail)

{

T = st[top++];

Visit(T);

if(T->lchild)

st[++tail] = T->lchild;

if(tail == TREE\_MAX\_SIZE)

exit(OVERFLOW);

if(T->rchild)

st[++tail] = T->rchild;

if(tail == TREE\_MAX\_SIZE)

exit(OVERFLOW);

}

return OK;

}

void Visit(BiTNode \*T){

if(flag == 0)

printf("二叉树未创建！\n");

else if(!T)

printf("二叉树为空！\n");

else

{

printf("%s\t",T->name);

printf("%s\n",T->tel);

}

}

BiTNode \*Find(BiTNode \*T, ElemType1 e[NAME\_MAX\_SIZE])

{

BiTNode \*N = NULL, \*P = NULL;

if(T == NULL)

return NULL;

if(!strcmp(T->name, e))

N = T;

P = Find(T->lchild, e);

if(P)

N = P;

P = Find(T->rchild, e);

if(P)

N = P;

return N;

}

status Output(BiTNode \*T, FILE \*fp){

if(flag == 0)

return SPEC;

if(T == NULL)

{

fprintf(fp, "#\n");

return ERROR;

}

fprintf(fp, "%s ", T->name);

if(strcmp(T->name, "#"))

fprintf(fp, "%s", T->tel);

fprintf(fp, "\n");

Output(T->lchild, fp);

Output(T->rchild, fp);

return OK;

}

status MenuInsert(Menu \*M, BiTNode \*T, char name[TREE\_NAME\_SIZE]){

MenuName \*newbase;

if(M->length >= M->menusize)

{

newbase = (MenuName\*)realloc(M->menuname, (++M->menusize) \* sizeof(MenuName));

if(!newbase)

exit(OVERFLOW);

M->menuname = newbase;

}

strcpy((M->menuname + M->length)->name, name);

(M->menuname + M->length)->bitree = T;

M->length++;

return OK;

}

status NewBiTree(BiTNode \*\*T){

\*T = (BiTNode \*)malloc(LENG);

(\*T)->lchild = NULL;

(\*T)->rchild = NULL;

flag = 0;

return OK;

}

int FindBiTree(Menu M, char name[TREE\_NAME\_SIZE]){

int i = 0;

for(;i < M.length; i++)

{

if(!strcmp(M.menuname[i].name, name))

return i;

}

return -1;

}

BiTNode \*ChooseBiTree(Menu M, char name[TREE\_NAME\_SIZE]){

int i = FindBiTree(M, name);

if(i == -1)

return NULL;

else

return M.menuname[i].bitree;

}

status MenuDelete(Menu \*M, char name[TREE\_NAME\_SIZE]){

int loc = FindBiTree(\*M, name);

if(loc == -1)

return ERROR;

DestroyBiTree(&(M->menuname[loc].bitree));

for(;loc < M->length - 1; loc++)

M->menuname[loc] = M->menuname[loc + 1];

M->length--;

return OK;

}

**附录D 基于邻接表的图实现的源程序**

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define SPEC -2

#define NONE -1

typedef int status;

typedef int ElemType\_Arc;

typedef int ElemType\_Ver;

#define MAX\_VERTEX\_NUM 20

#define MAX\_ARC\_NUM 200

#define MAX\_ARC\_NAME 20

#define MAX\_VERTEX\_NAME 20

#define GRAPH\_NAME\_SIZE 20

#define MENU\_INIT\_SIZE 50

int visited[50];

static char menu\_name[GRAPH\_NAME\_SIZE];

typedef struct ArcInfo{

char arc\_name[MAX\_ARC\_NAME];

ElemType\_Arc arc\_info;

}ArcInfo;

typedef struct VInfo{

char ver\_name[MAX\_VERTEX\_NAME];

ElemType\_Ver ver\_info;

}VInfo;

typedef struct ArcNode{//表结点结构类型

int adjvex; //该弧(边)的终点位置

struct ArcNode \*nextarc; //指向下一条弧的指针

ArcInfo \*info; //该弧的相关信息的指针

} ArcNode;

typedef struct VNode {//头结点的类型

VInfo data; //顶点信息

ArcNode \*firstarc; //指向第一条弧

}VNode, AdjList;

typedef struct {//邻接表

AdjList \*vertices;

int vexnum, arcnum; //图中顶点数n和边数e

int kind; //图的类型

} ALGraph;

typedef struct {

int adjvex\_start;

int adjvex\_end;

ArcInfo arc\_data;

}ArcCreateInfo;

typedef struct MenuName

{

char name[GRAPH\_NAME\_SIZE];

ALGraph graph;

}MenuName;

typedef struct Menu{

MenuName \*menuname;

int length;

int menusize;

}Menu;

status CreateGraph(ALGraph \*G,VInfo \*V,ArcCreateInfo \*VR);

status DestroyGraph(ALGraph \*G);

int LocateVex(ALGraph G, int u);

ElemType\_Ver GetVex(ALGraph G, int v);

status PutVex(ALGraph \*G, int v, ElemType\_Ver value);

int FirstAdjVex(ALGraph G, int v);

int NextAdjVex(ALGraph G, int v, int w);

status InsertVex(ALGraph \*G, VInfo V);

status DeleteVex(ALGraph \*G, int v);

status InsertArc(ALGraph \*G, int v, int w);

status DeleteArc(ALGraph \*G, int v, int w);

void DFSTraverse(ALGraph G, void (\*Visit) (VNode V));

void DFS(ALGraph G, int v, void (\*Visit) (VNode V));

void BFSTraverse(ALGraph G, void (\*Visit) (VNode V));

void Visit(VNode V);

status FindArcNode(ALGraph G, int v, int w);

status MenuInsert(Menu \*M, ALGraph G, char name[GRAPH\_NAME\_SIZE]);

status NewGraph(ALGraph \*G);

int FindGraph(Menu M, char name[GRAPH\_NAME\_SIZE]);

ALGraph ChooseGraph(Menu M, char name[GRAPH\_NAME\_SIZE]);

status MenuDelete(Menu \*M, char name[GRAPH\_NAME\_SIZE]);

void main(void){

int i = 1, op = 1;

FILE \*fp;

ALGraph G;

G.vertices = (AdjList \*)malloc((MAX\_VERTEX\_NUM + 1) \* sizeof(AdjList));

for(;i <= MAX\_VERTEX\_NUM;i++)

{

G.vertices[i].firstarc = NULL;

strcpy(G.vertices[i].data.ver\_name, "#");

G.vertices[i].data.ver\_info = 0;

}

VInfo V[MAX\_VERTEX\_NUM + 2];

ArcCreateInfo VR[MAX\_ARC\_NUM + 2];

char graph\_name[GRAPH\_NAME\_SIZE];

G.arcnum = 0;

G.vexnum = 0;

G.kind = 0;

Menu M;

M.menuname = (MenuName \*)malloc(MENU\_INIT\_SIZE \* sizeof(MenuName));

M.length = 0;

M.menusize = MENU\_INIT\_SIZE;

while(op)

{

ALGraph L;

int e = 1,re = 0, u = 0, v = 0, value = 0, w = 0;

ElemType\_Ver re\_ver;

VInfo VE;

int ver\_num;

char ver\_name[MAX\_VERTEX\_NAME];

ArcNode \*N;

ALGraph A;

system("cls"); printf("\n\n");

printf(" Menu for Binary Tree On Sequence Structure \n");

printf("-------------------------------------------------\n");

printf(" 1. CreateGraph 8. InsertVex\n");

printf(" 2. DestroyGraph 9. DeleteVex\n");

printf(" 3. LocateVex 10. InsertArc \n");

printf(" 4. GetVex 11. DeleteArc\n");

printf(" 5. PutVex 12. DFSTraverse\n");

printf(" 6. FirstAdjVex 13. BFSTraverse\n");

printf(" 7. NextAdjVex\n");

printf("-------------------------------------------------\n");

printf(" 14. DataSave 17. MenuDelete\n");

printf(" 15. DataLoad 18. NewGraph\n");

printf(" 16. MenuInsert 19. ChooseGraph\n");

printf(" 0. Exit\n");

printf("-------------------------------------------------\n");

printf(" 请选择你的操作[0~19]:");

scanf("%d",&op);

switch(op){

case 1:

//CreateGraph

printf("下面录入顶点集的信息：（输入‘#’结束）\n");

while(1)

{

if(e >= MAX\_VERTEX\_NUM)

break;

printf("请输入顶点的名称：\n");

scanf("%s", V[e].ver\_name);

if(!strcmp(V[e].ver\_name, "#"))

break;

printf("请输入顶点的元素值：\n");

scanf("%d", &V[e].ver\_info);

printf("\n");

e++;

}

e = 1;

printf("下面录入弧集的信息：（输入‘0’结束）\n");

while(1)

{

if(e >= MAX\_ARC\_NUM)

break;

printf("请输入弧的起点：\n");

scanf("%d", &VR[e].adjvex\_start);

if(VR[e].adjvex\_start == 0)

break;

printf("请输入弧的终点：\n");

scanf("%d", &VR[e].adjvex\_end);

printf("请输入弧的名称：\n");

scanf("%s", VR[e].arc\_data.arc\_name);

printf("请输入弧的值：\n");

scanf("%d", &VR[e].arc\_data.arc\_info);

printf("\n");

e++;

}

if(CreateGraph(&G, V, VR))

printf("创建成功！\n");

else

printf("创建失败！\n");

getchar();

getchar();

break;

case 2:

//DestroyGraph

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

DestroyGraph(&G);

printf("销毁成功！\n");

getchar();

getchar();

break;

case 3:

//LocateVex

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的顶点元素值：\n");

scanf("%d", &u);

re = LocateVex(G, u);

if(re == ERROR)

printf("未找到相应的顶点！\n");

else

printf("所查询的顶点的位置为%d\n", re);

getchar();

getchar();

break;

case 4:

//GetVex

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的顶点的位置：\n");

scanf("%d", &v);

re\_ver = GetVex(G, v);

if(re\_ver == ERROR)

printf("位置输入错误！\n");

else

printf("查询的顶点的元素值为%d", re\_ver);

getchar();

getchar();

break;

case 5:

//PutVex

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要赋值的顶点的位置：\n");

scanf("%d", &v);

printf("请输入要赋给顶点的值：\n");

scanf("%d", &value);

if(PutVex(&G, v, value) == ERROR)

printf("位置输入错误！\n");

else

printf("赋值成功！\n");

getchar();

getchar();

break;

case 6:

//FirstAdjVex

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的顶点的位置：\n");

scanf("%d", &v);

re = FirstAdjVex(G, v);

if(re == NONE)

printf("该顶点没有邻接点！\n");

else if(re == ERROR)

printf("位置输入错误！\n");

else

printf("该顶点邻接点的位置为%d", re);

getchar();

getchar();

break;

case 7:

//NextAdjVex

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要查询的顶点的位置：\n");

scanf("%d", &v);

printf("请输入当前邻接点的位置：\n");

scanf("%d", &w);

re = NextAdjVex(G, v, w);

if(re == NONE)

printf("该邻接点为最后一个邻接点！\n");

else if(re == ERROR)

printf("位置输入错误！\n");

else if(re == SPEC)

printf("未找到相应的邻接点！\n");

else

printf("下一个邻接点的位置为%d", re);

getchar();

getchar();

break;

case 8:

//InsertVex

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要插入的顶点的名称：\n");

scanf("%s", VE.ver\_name);

printf("请输入要插入的顶点的值：\n");

scanf("%d", &VE.ver\_info);

if(InsertVex(&G, VE) == ERROR)

printf("顶点插入失败！\n");

else

printf("顶点插入成功！\n");

getchar();

getchar();

break;

case 9:

//DeleteVex

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要删除的结点的位置：\n");

scanf("%d", &v);

if(DeleteVex(&G, v) == ERROR)

printf("位置输入错误！\n");

else

printf("顶点删除成功！\n");

getchar();

getchar();

break;

case 10:

//InsertArc

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要插入的弧的起点：\n");

scanf("%d", &v);

printf("请输入要插入的弧的终点：\n");

scanf("%d", &w);

if(InsertArc(&G, v, w) == OK)

printf("弧插入成功！\n");

else

printf("弧插入失败！\n");

getchar();

getchar();

break;

case 11:

//DeleteArc

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入要删除的弧的起点：\n");

scanf("%d", &v);

printf("请输入要删除的弧的终点：\n");

scanf("%d", &w);

if(DeleteArc(&G, v, w) == OK)

printf("弧删除成功！\n");

else

printf("弧删除失败！\n");

getchar();

getchar();

break;

case 12:

//DFSTraverse

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

DFSTraverse(G, Visit);

getchar();

getchar();

break;

case 13:

//BFSTraverse

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

BFSTraverse(G, Visit);

getchar();

getchar();

break;

case 14:

//DataSave

if((fp = fopen("test.txt","w")) == NULL)

{

printf("文件无法打开!\n");

getchar();

getchar();

break;

}

fprintf(fp, "%d\n", G.vexnum);

for(e = 1; e <= G.vexnum; e++)

{

fprintf(fp, "%s ", G.vertices[e].data.ver\_name);

fprintf(fp, "%d\n", G.vertices[e].data.ver\_info);

}

fprintf(fp, "#\n");

for(e = 1; e <= G.vexnum; e++)

{

N = G.vertices[e].firstarc;

while(N != NULL)

{

fprintf(fp, "%d ", e);

fprintf(fp, "%d ", N->adjvex);

fprintf(fp, "%s ", N->info->arc\_name);

fprintf(fp, "%d\n", N->info->arc\_info);

N = N->nextarc;

}

}

printf("文件保存成功！\n");

fclose(fp);

getchar();

getchar();

break;

case 15:

//DataLoad

if((fp = fopen("test.txt","r")) == NULL)

{

printf("文件无法打开!\n");

getchar();

getchar();

break;

}

fscanf(fp, "%d", &ver\_num);

for(e = 1; e <= ver\_num; e++)

{

fscanf(fp, "%s", V[e].ver\_name);

fscanf(fp, "%d", &V[e].ver\_info);

}

fscanf(fp, "%s", V[e].ver\_name);

e = 1;

while(fscanf(fp, "%d", &VR[e].adjvex\_start) != EOF)

{

fscanf(fp, "%d", &VR[e].adjvex\_end);

fscanf(fp, "%s", VR[e].arc\_data.arc\_name);

fscanf(fp, "%d", &VR[e].arc\_data.arc\_info);

e++;

}

VR[e].adjvex\_start = 0;

CreateGraph(&G, V, VR);

printf("文件读取成功！\n");

fclose(fp);

getchar();

getchar();

break;

case 16:

//MenuInsert

if(G.vexnum == 0)

{

printf("图未创建！\n");

getchar();

getchar();

break;

}

printf("请输入图的名称！\n");

scanf("%s", graph\_name);

MenuInsert(&M, G, graph\_name);

printf("图插入成功！\n");

strcpy(menu\_name, graph\_name);

getchar();

getchar();

break;

case 17:

//MenuDelete

printf("请输入要删除的图的名称\n");

scanf("%s", graph\_name);

if(MenuDelete(&M, graph\_name) == ERROR)

printf("未找到相应名称的图！\n");

else

printf("图删除成功！\n");

getchar();

getchar();

break;

case 18:

//NewGraph

NewGraph(&L);

G = L;

printf("已创建新的图！\n");

getchar();

getchar();

break;

case 19:

//ChooseGraph

printf("请输入要操作的图的名称！\n");

scanf("%s", graph\_name);

A = ChooseGraph(M, graph\_name);

if(FindGraph(M, graph\_name) == -1)

printf("未找到相应名称的图！\n");

else

{

G = A;

strcpy(menu\_name, graph\_name);

printf("已成功找到图，请进行操作！\n");

}

getchar();

getchar();

break;

}

}

}

status CreateGraph(ALGraph \*G,VInfo \*V,ArcCreateInfo \*VR){

int i = 1;

while(strcmp(V[i].ver\_name, "#"))

{

strcpy(G->vertices[i].data.ver\_name, V[i].ver\_name);

G->vertices[i].data.ver\_info = V[i].ver\_info;

G->vexnum++;

i++;

}

for(i = 1; VR[i].adjvex\_start != 0; i++)

{

if(FindArcNode(\*G, VR[i].adjvex\_end, VR[i].adjvex\_start))

continue;

ArcNode \*A = (ArcNode \*)malloc(sizeof(ArcNode));

if(!A)

exit(OVERFLOW);

ArcInfo \*AI = (ArcInfo \*)malloc(sizeof(ArcInfo));

if(!AI)

exit(OVERFLOW);

A->adjvex = VR[i].adjvex\_end;

A->nextarc = NULL;

AI->arc\_info = VR[i].arc\_data.arc\_info;

strcpy(AI->arc\_name, VR[i].arc\_data.arc\_name);

A->info = AI;

if(G->vertices[VR[i].adjvex\_start].firstarc == NULL)

G->vertices[VR[i].adjvex\_start].firstarc = A;

else

{

ArcNode \*N = G->vertices[VR[i].adjvex\_start].firstarc;

while(N->nextarc != NULL)

N = N->nextarc;

N->nextarc = A;

}

ArcNode \*B = (ArcNode \*)malloc(sizeof(ArcNode));

if(!B)

exit(OVERFLOW);

ArcInfo \*BI = (ArcInfo \*)malloc(sizeof(ArcInfo));

if(!BI)

exit(OVERFLOW);

B->adjvex = VR[i].adjvex\_start;

B->nextarc = NULL;

BI->arc\_info = VR[i].arc\_data.arc\_info;

strcpy(BI->arc\_name, VR[i].arc\_data.arc\_name);

B->info = BI;

if(G->vertices[VR[i].adjvex\_end].firstarc == NULL)

G->vertices[VR[i].adjvex\_end].firstarc = B;

else

{

ArcNode \*P = G->vertices[VR[i].adjvex\_end].firstarc;

while(P->nextarc != NULL)

P = P->nextarc;

P->nextarc = B;

}

G->arcnum = G->arcnum + 2;

}

return OK;

}

status DestroyGraph(ALGraph \*G){

int i = 1;

ArcNode \*P, \*N;

for(i = 1;i <= G->vexnum; i++)

{

if(G->vertices[i].firstarc != NULL)

{

P = G->vertices[i].firstarc;

N = G->vertices[i].firstarc->nextarc;

while(N != NULL)

{

if(P->info != NULL)

free(P->info);

P->info = NULL;

free(P);

P = N;

N = N->nextarc;

}

free(P);

G->vertices[i].firstarc = NULL;

}

}

G->arcnum = 0;

G->vexnum = 0;

return OK;

}

int LocateVex(ALGraph G, int u){

int i;

if(u == 0)

return ERROR;

for(i = 1;i <= G.vexnum; i++)

{

if(G.vertices[i].data.ver\_info == u)

return i;

}

return ERROR;

}

ElemType\_Ver GetVex(ALGraph G, int v){

if(v <= 0 || v > G.vexnum)

return ERROR;

else if(G.vertices[v].data.ver\_info == 0)

return ERROR;

else

return G.vertices[v].data.ver\_info;

}

status PutVex(ALGraph \*G, int v, ElemType\_Ver value){

if(v <= 0 || v > G->vexnum)

return ERROR;

else if(G->vertices[v].data.ver\_info == 0)

return ERROR;

else

{

G->vertices[v].data.ver\_info = value;

return OK;

}

}

int FirstAdjVex(ALGraph G, int v){

if(v <= 0 || v > G.vexnum)

return ERROR;

else

{

if(G.vertices[v].firstarc == NULL)

return NONE;

else

return G.vertices[v].firstarc->adjvex;

}

}

int NextAdjVex(ALGraph G, int v, int w){

ArcNode \*N = NULL;

if(v <= 0 || v > G.vexnum)

return ERROR;

else

{

N = G.vertices[v].firstarc;

while(1)

{

if(N == NULL)

return SPEC;

if(N->adjvex == w)

break;

N = N->nextarc;

}

if(N->nextarc == NULL)

return NONE;

else

return N->nextarc->adjvex;

}

}

status InsertVex(ALGraph \*G, VInfo V){

if(G->vexnum >= MAX\_VERTEX\_NUM)

{

AdjList \*newbase;

newbase = (AdjList \*)realloc(G->vertices, (G->vexnum + 2) \* sizeof(AdjList));

if(newbase == NULL)

return ERROR;

G->vertices = newbase;

}

G->vexnum++;

strcpy(G->vertices[G->vexnum].data.ver\_name, V.ver\_name);

G->vertices[G->vexnum].data.ver\_info = V.ver\_info;

G->vertices[G->vexnum].firstarc = NULL;

return OK;

}

status DeleteVex(ALGraph \*G, int v){

int i = 1;

if(v <= 0 || v > G->vexnum)

return ERROR;

ArcNode \*N, \*P;

if(G->vertices[v].firstarc != NULL)

{

P = G->vertices[v].firstarc;

N = G->vertices[v].firstarc->nextarc;

while(N != NULL)

{

free(P->info);

P->info = NULL;

free(P);

P = N;

N = N->nextarc;

G->arcnum--;

}

free(P);

}

G->vertices[v].firstarc = NULL;

G->vertices[v].data.ver\_info = 0;

strcpy(G->vertices[v].data.ver\_name, "#");

for(i = 1; i < G->vexnum; i++)

{

if(G->vertices[i].firstarc != NULL)

{

P = G->vertices[i].firstarc;

N = G->vertices[i].firstarc->nextarc;

if(P->adjvex == v)

{

free(P);

P = NULL;

if(N == NULL)

G->vertices[i].firstarc = NULL;

else

G->vertices[i].firstarc = N;

G->arcnum--;

}

else

{

while(N != NULL)

{

if(N->adjvex == v)

{

P->nextarc = N->nextarc;

free(N);

N = NULL;

G->arcnum--;

break;

}

P = P->nextarc;

N = N->nextarc;

}

}

}

}

return OK;

}

status InsertArc(ALGraph \*G, int v, int w){

int arc\_info;

char arc\_name[MAX\_ARC\_NAME];

if(v <= 0 || v > G->vexnum)

return ERROR;

if(w <= 0 || w > G->vexnum)

return ERROR;

ArcNode \*N;

ArcNode \*A = (ArcNode \*)malloc(sizeof(ArcNode));

if(!A)

exit(OVERFLOW);

A->adjvex = w;

A->nextarc = NULL;

A->info = (ArcInfo \*)malloc(sizeof(ArcInfo));

if(!A->info)

exit(OVERFLOW);

printf("请输入弧的名称：\n");

scanf("%s",arc\_name);

printf("请输入弧的值：\n");

scanf("%d",&arc\_info);

strcpy(A->info->arc\_name, arc\_name);

A->info->arc\_info = arc\_info;

N = G->vertices[v].firstarc;

if(N == NULL)

G->vertices[v].firstarc = A;

else

{

while(N->nextarc != NULL)

N = N->nextarc;

N->nextarc = A;

}

ArcNode \*B = (ArcNode \*)malloc(sizeof(ArcNode));

if(!B)

exit(OVERFLOW);

B->adjvex = v;

B->nextarc = NULL;

B->info = (ArcInfo \*)malloc(sizeof(ArcInfo));

if(!B->info)

exit(OVERFLOW);

strcpy(B->info->arc\_name, arc\_name);

B->info->arc\_info = arc\_info;

N = G->vertices[w].firstarc;

if(N == NULL)

G->vertices[w].firstarc = B;

else

{

while(N->nextarc != NULL)

N = N->nextarc;

N->nextarc = B;

}

G->arcnum = G->arcnum + 2;

return OK;

}

status DeleteArc(ALGraph \*G, int v, int w){

ArcNode \*N, \*P;

int flag = 0;

if(G->vertices[v].firstarc != NULL)

{

P = G->vertices[v].firstarc;

N = G->vertices[v].firstarc->nextarc;

if(P->adjvex == w)

{

free(P);

P = NULL;

if(N == NULL)

G->vertices[v].firstarc = NULL;

else

G->vertices[v].firstarc = N;

G->arcnum--;

flag = 1;

}

else

{

while(N != NULL)

{

if(N->adjvex == w)

{

P->nextarc = N->nextarc;

free(N);

N = NULL;

G->arcnum--;

flag = 1;

break;

}

P = P->nextarc;

N = N->nextarc;

}

}

}

if(!flag)

return ERROR;

P = G->vertices[w].firstarc;

N = G->vertices[w].firstarc->nextarc;

if(P->adjvex == v)

{

free(P);

P = NULL;

if(N == NULL)

G->vertices[w].firstarc = NULL;

else

G->vertices[w].firstarc = N;

G->arcnum--;

}

else

{

while(N != NULL)

{

if(N->adjvex == v)

{

P->nextarc = N->nextarc;

free(N);

N = NULL;

G->arcnum--;

break;

}

P = P->nextarc;

N = N->nextarc;

}

}

return OK;

}

void DFSTraverse(ALGraph G, void (\*Visit) (VNode V)){

int i;

for(i = 1;i <= G.vexnum; i++)

visited[i] = 0;

for(i = 1;i <= G.vexnum; i++)

{

if(!visited[i])

DFS(G, i, Visit);

}

printf("以上为图的深度优先遍历！\n");

}

void DFS(ALGraph G, int v, void (\*Visit) (VNode V)){

int w;

visited[v] = 1;

Visit(G.vertices[v]);

for(w = FirstAdjVex(G, v); w >= 1; w = NextAdjVex(G, v, w))

{

if(!visited[w])

DFS(G, w, Visit);

}

}

void BFSTraverse(ALGraph G, void (\*Visit) (VNode V)){

int i, u, w;

int Q[G.vexnum + 1];

ArcNode \*N = G.vertices[i].firstarc;

for(i = 1;i <= G.vexnum; i++)

visited[i] = 0;

for(i = 1;i <= G.vexnum; i++)

{

int top = 1, tail = 1;

if(!visited[i])

{

visited[i] = 1;

Visit(G.vertices[i]);

Q[tail] = i;

while(top <= tail)

{

u = Q[top++];

for(w = FirstAdjVex(G, u); w >= 1; w = NextAdjVex(G, u, w))

{

if(!visited[w])

{

visited[w] = 1;

Visit(G.vertices[w]);

Q[++tail] = w;

}

}

}

}

}

printf("以上为图的广度优先遍历！\n");

}

void Visit(VNode V){

if(strcmp(V.data.ver\_name, "#"))

printf("结点名称为：%s\n", V.data.ver\_name);

if(V.data.ver\_info != 0)

printf("结点值为：%d\n", V.data.ver\_info);

}

status FindArcNode(ALGraph G, int v, int w){

ArcNode \*N = G.vertices[v].firstarc;

while(N != NULL)

{

if(N->adjvex == w)

return OK;

N = N->nextarc;

}

return ERROR;

}

status MenuInsert(Menu \*M, ALGraph G, char name[GRAPH\_NAME\_SIZE]){

MenuName \*newbase;

if(M->length >= M->menusize)

{

newbase = (MenuName\*)realloc(M->menuname, (++M->menusize) \* sizeof(MenuName));

if(!newbase)

exit(OVERFLOW);

M->menuname = newbase;

}

strcpy((M->menuname + M->length)->name, name);

(M->menuname + M->length)->graph = G;

M->length++;

return OK;

}

status NewGraph(ALGraph \*G){

int i = 1;

G->vertices = (AdjList \*)malloc((MAX\_VERTEX\_NUM + 1) \* sizeof(AdjList));

for(i = 1;i <= MAX\_VERTEX\_NUM;i++)

{

G->vertices[i].firstarc = NULL;

strcpy(G->vertices[i].data.ver\_name, "#");

G->vertices[i].data.ver\_info = 0;

}

G->arcnum = 0;

G->vexnum = 0;

G->kind = 0;

return OK;

}

int FindGraph(Menu M, char name[GRAPH\_NAME\_SIZE]){

int i = 0;

for(;i < M.length; i++)

{

if(!strcmp(M.menuname[i].name, name))

return i;

}

return -1;

}

ALGraph ChooseGraph(Menu M, char name[GRAPH\_NAME\_SIZE]){

int i = FindGraph(M, name);

ALGraph G;

G.vexnum = 0;

if(i == -1)

return G;

else

return M.menuname[i].graph;

}

status MenuDelete(Menu \*M, char name[GRAPH\_NAME\_SIZE]){

int loc = FindGraph(\*M, name);

if(loc == -1)

return ERROR;

DestroyGraph(&M->menuname[loc].graph);

for(;loc < M->length - 1; loc++)

M->menuname[loc] = M->menuname[loc + 1];

M->length--;

return OK;

}