

**数据结构实验**

**(2023级计科、计卓、本硕博各专业班级)**

**计算机科学与技术学院**

**数组结构课程组**

**2024年春季**

目录

[1基于顺序存储结构的线性表实现 2](#_Toc9569)

[1.1 实验目的 2](#_Toc3588)

[1.2 线性表运算定义 2](#_Toc11377)

[1.3 实验任务 3](#_Toc10905)

[2基于链式存储结构的线性表实现 4](#_Toc15494)

[2.1 实验目的 4](#_Toc22401)

[2.2 线性表运算定义 4](#_Toc29306)

[2.3 实验任务 5](#_Toc833)

[3基于二叉链表的二叉树实现 6](#_Toc31628)

[3.1 实验目的 6](#_Toc25532)

[3.2 二叉树运算定义 6](#_Toc1319)

[3.3 实验任务 8](#_Toc27492)

[4基于邻接表的图实现 9](#_Toc4066)

[4.1 实验目的 9](#_Toc14518)

[4.2 图基本运算定义 9](#_Toc26498)

[4.3 实验任务 11](#_Toc23222)

[参考文献 12](#_Toc20482)

[附录A顺序表实现框架程序清单 13](#_Toc23123)

[附录B数据元素的文件读写 16](#_Toc14321)

[附录C线性表模板实例 17](#_Toc14292)

实验说明

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

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

**线上**平台网址：<https://www.educoder.net/>，各位同学实名注册后，使用各自班级的邀请码，以学生身份加入《数据结构实验2023春季》课堂。各班的邀请码：

CS2301：KD9P3T CS2302：LSVCZP

CS2303：Z3GWLN CS2304：HRJDMW

CS2305：4NDW9U CS2306：PA3H42

CS2307：A5C8M3 CS2308：TC8VEM

CS2309：AEL48Q CS2310：FRV4NY

CS2311：NL9ZEF

计卓2301：39DHTX 本硕博2301：L4NXA9

**注**：线上提交实验作业均严格按照系统设置时间。

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

**线下**检查：每个同学最多两次面对面检查机会。检查期间根据所提供的测试用例认真检查后，记录在“《数据结构实验》功能检查记载表”上。如有余力的同学可完成任务书提供的附加功能，整合在演示系统中，并根据检查情况进行记录。相关代码规范（命名、注释、排版等）一并检查。（功能检查70%）

**实验报告说明**：（报告占30%）

各位同学根据每个实验的完成情况，按照学院提供的“华中科技大学计算机学院实验报告Latex模板（数据结构）”撰写实验报告。模板中四个实验分为四章进行撰写，同学们自行选择完成**两章**即可：第一二个实验（顺序表、链表）选择其中一个实验完成报告设为第一章；第三四个实验（二叉树、图）选择其中一个实验完成报告设为第二章。但报告中的附录**必须**包含四个实验的源代码。

**最后材料提交：**

演示系统的源程序应按照代码规范增加注释和排版，目标程序务必是可以独立于IDE运行的EXE文件。根据指导老师公布的时间提交纸质报告（双面打印，可不包含源代码）和相关电子资料（主要包括实验报告、实验源程序和实验目标程序等）。所有电子资料均存储于每位同学自己的相应文件夹下，其文件夹名称格式为“专业班级\_学号\_姓名”，然后根据不同实验分子文件夹（顺序表、链表、二叉树、图）存放相应资料，由各班学委收齐后交给相关指导教师。

# 1基于顺序存储结构的线性表实现

## 1.1 实验目的

通过实验达到：（1）加深对线性表的概念、基本运算的理解；（2）熟练掌握线性表的逻辑结构与物理结构的关系；（3）物理结构采用顺序表，熟练掌握顺序表基本运算的实现。

## 1.2 线性表运算定义

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：

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

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

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

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

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

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

（7）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；

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

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

（10）插入元素：函数名称是ListInsert(L,i,e)；初始条件是线性表L已存在，1≤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。

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

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

附加功能：

（1）最大连续子数组和：函数名称是MaxSubArray(L); 初始条件是线性表L已存在且非空，请找出一个具有最大和的连续子数组（子数组最少包含一个元素），操作结果是其最大和；

（2）[和为K的子数组](https://leetcode-cn.com/problems/subarray-sum-equals-k/)：函数名称是SubArrayNum(L,k); 初始条件是线性表L已存在且非空, 操作结果是该数组中和为k的连续子数组的个数；

（3）顺序表排序：函数名称是sortList(L)；初始条件是线性表L已存在；操作结果是将L由小到大排序；

（4）实现线性表的文件形式保存：其中，①需要设计文件数据记录格式，以高效保存线性表数据逻辑结构(D,{R})的完整信息；②需要设计线性表文件保存和加载操作合理模式。附录B提供了文件存取的参考方法。

注：①保存到文件后，可以由一个空线性表加载文件生成线性表。

（5）实现多个线性表管理：设计相应的数据结构管理多个线性表的查找、添加、移除等功能。

注：附加功能（5）实现多个线性表管理应能创建、添加、移除多个线性表，单线性表和多线性表的区别仅仅在于线性表管理的个数不同，多线性表管理应可自由切换到管理的每个表，并可单独对某线性表进行单线性表的所有操作。

## 1.3 实验任务

采用顺序表作为线性表的物理结构，实现1.2小节的运算。其中ElemType为数据元素的类型名，具体含义可自行定义，其它有关类型和常量的定义和引用详见文献[1]的p10。

构造一个具有菜单的功能演示系统。其中，在主程序中完成函数调用所需实参值的准备和函数执行结果的显示，并给出适当的操作提示显示。附录A提供了简易菜单的框架（供参考）。

# 2基于链式存储结构的线性表实现

## 2.1 实验目的

通过实验达到：（1）加深对线性表的概念、基本运算的理解；（2）熟练掌握线性表的逻辑结构与物理结构的关系；（3）物理结构采用单链表,熟练掌握线性表的基本运算的实现。

## 2.2 线性表运算定义

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：

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

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

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

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

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

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

（7）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；

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

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

（10）插入元素：函数名称是ListInsert(L,i,e)；初始条件是线性表L已存在，1≤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；

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

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

附加功能：

（1）链表翻转：函数名称是reverseList(L)，初始条件是线性表L已存在；操作结果是将L翻转；

（2）删除链表的倒数第n个结点：函数名称是RemoveNthFromEnd(L,n); 初始条件是线性表L已存在且非空, 操作结果是该链表中倒数第n个节点；

（3）链表排序：函数名称是sortList(L)，初始条件是线性表L已存在；操作结果是将L由小到大排序；

（4）实现线性表的文件形式保存：其中，①需要设计文件数据记录格式，以高效保存线性表数据逻辑结构(D,{R})的完整信息；②需要设计线性表文件保存和加载操作合理模式。附录B提供了文件存取的参考方法；

注：①保存到文件后，可以由一个空表加载文件生成单链表。

（5）实现多个线性表管理：设计相应的数据结构管理多个线性表的查找、添加、移除等功能。

注：附加功能（5）实现多个线性表管理应能创建、添加、移除多个线性表，单线性表和多线性表的区别仅仅在于线性表管理的个数不同，多线性表管理应可自由切换到管理的每个表，并可单独对某线性表进行单线性表的所有操作。

## 2.3 实验任务

采用单链表作为线性表的物理结构，实现2.2小节的运算。其中ElemType为数据元素的类型名，具体含义可自行定义，其它有关类型和常量的定义和引用详见文献[1]的p10。

构造一个具有菜单的功能演示系统。其中，在主程序中完成函数调用所需实参值的准备和函数执行结果的显示，并给出适当的操作提示显示。附录A提供了简易菜单的框架（供参考）。

# 3基于二叉链表的二叉树实现

## 3.1 实验目的

通过实验达到：（1）加深对二叉树的概念、基本运算的理解；（2）熟练掌握二叉树的逻辑结构与物理结构的关系；（3）以二叉链表作为物理结构，熟练掌握二叉树基本运算的实现。

## 3.2 二叉树运算定义

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创建二叉树、销毁二叉树、清空二叉树、判定空二叉树和求二叉树深度等14种基本运算。具体运算功能定义和说明如下：

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

**注：**①要求T中各结点关键字具有唯一性。后面各操作的实现，也都要满足一棵二叉树中关键字的唯一性，不再赘述；②CreateBiTree中根据definition生成T，不应在CreateBiTree中输入二叉树的定义。

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

**注：**①销毁了则原先所有分配的空间都销毁。

1. 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；

**注：**①清空只是将二叉树清空了，是空树，但是在内存中还存在空间，只是值不确定。

（4）判定空二叉树：函数名称是BiTreeEmpty(T)；初始条件是二叉树T存在；操作结果是若T为空二叉树则返回TRUE，否则返回FALSE；

（5）求二叉树深度：函数名称是BiTreeDepth(T)；初始条件是二叉树T存在；操作结果是返回T的深度；

（6）查找结点：函数名称是LocateNode(T,e)；初始条件是二叉树T已存在，e是和T中结点关键字类型相同的给定值；操作结果是返回查找到的结点指针，如无关键字为e的结点，返回NULL；

（7）结点赋值：函数名称是Assign(T,e,value)；初始条件是二叉树T已存在，e是和T中结点关键字类型相同的给定值；操作结果是关键字为e的结点赋值为value；

（8）获得兄弟结点：函数名称是GetSibling(T,e)；初始条件是二叉树T存在，e是和T中结点关键字类型相同的给定值；操作结果是返回关键字为e的结点的（左或右）兄弟结点指针。若关键字为e的结点无兄弟，则返回NULL；

（9）插入结点：函数名称是InsertNode(T,e,LR,c)；初始条件是二叉树T存在，e是和T中结点关键字类型相同的给定值，LR为0或1，c是待插入结点；操作结果是根据LR为0或者1，插入结点c到T中，作为关键字为e的结点的左或右孩子结点，结点e的原有左子树或右子树则为结点c的右子树；

特殊情况，c插入作为根结点？可以考虑LR为-1时，作为根结点插入，原根结点作为c的右子树。

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

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

**注：**前序、中序和后序三种遍历算法，要求至少一个用非递归算法实现。

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

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

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

附加功能：

1. 最大路径和：函数名称是MaxPathSum(T)，初始条件是二叉树T存在；操作结果是返回根节点到叶子结点的最大路径和；

注：①从根节点开始。

1. [最近公共祖先](https://leetcode-cn.com/problems/lowest-common-ancestor-of-a-binary-tree/)：函数名称是LowestCommonAncestor(T,e1,e2)；初始条件是二叉树T存在；操作结果是该二叉树中e1节点和e2节点的最近公共祖先；

注：①最近公共祖先可以是e1，e2中的某个节点。

（3）[翻转二叉树](https://leetcode-cn.com/problems/invert-binary-tree/)：函数名称是InvertTree(T)，初始条件是线性表L已存在；操作结果是将T翻转，使其所有节点的左右节点互换；

（4）实现线性表的文件形式保存：其中，①需要设计文件数据记录格式，以高效保存二叉树数据逻辑结构(D,{R})的完整信息；②需要设计二叉树文件保存和加载操作合理模式。附录B提供了文件存取的方法；

注：①保存到文件后，可以由一个空二叉树加载文件生成二叉树。

（5）实现多个二叉树管理：可采用线性表的方式管理多个二叉树，线性表中的每个数据元素为一个二叉树的基本属性，至少应包含有二叉树的名称。基于顺序表实现的二叉树管理，其物理结构的参考设计如图3-1所示。



图3-1多二叉树管理的物理结构示意图

注：附加功能（5）实现多个二叉树管理应能创建、添加、移除多个二叉树，单二叉树和多二叉树的区别仅仅在于二叉树管理的个数不同，多二叉树管理应可自由切换到管理的每个二叉树，并可单独对某二叉树进行单二叉树的所有操作。

## 3.3 实验任务

采用二叉链表表作为二叉树的物理结构，实现3.2小节的运算。其中ElemType为数据元素的类型名，具体含义可自行定义，但要求二叉树结点类型为结构型，至少包含二个部分，一个是能唯一标识一个结点的关键字（类似于学号或职工号），另一个是其它部分。其它有关类型和常量的定义和引用详见文献[1]的p10。

构造一个具有菜单的功能演示系统。其中，在主程序中完成函数调用所需实参值的准备和函数执行结果的显示，并给出适当的操作提示显示（供参考）。

# 4基于邻接表的图实现

## 4.1 实验目的

通过实验达到：（1）加深对图的概念、基本运算的理解；（2）熟练掌握图的逻辑结构与物理结构的关系；（3）以邻接表作为物理结构，熟练掌握图基本运算的实现。

## 4.2 图基本运算定义

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12种基本运算。具体运算功能定义和说明如下。具体运算功能定义如下：

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

注：①要求图G中顶点关键字具有唯一性。后面各操作的实现，也都要满足一个图中关键字的唯一性，不再赘述；② V和VR对应的是图的逻辑定义形式，比如V为顶点序列，VR为关键字对的序列。不能将邻接矩阵等物理结构来代替V和VR。

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

（3）查找顶点：函数名称是LocateVex(G,u)；初始条件是图G存在，u是和G中顶点关键字类型相同的给定值；操作结果是若u在图G中存在，返回关键字为u的顶点位置序号（简称位序），否则返回其它表示“不存在”的信息；

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

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

（6）获得下一邻接点：函数名称是NextAdjVex(G, v, w)；初始条件是图G存在，v和w是G中两个顶点的位序，v对应G的一个顶点,w对应v的邻接顶点；操作结果是返回v的（相对于w）下一个邻接顶点的位序，如果w是最后一个邻接顶点，返回其它表示“不存在”的信息；

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

（8）删除顶点：函数名称是DeleteVex(G,v)；初始条件是图G存在，v是和G中顶点关键字类型相同的给定值；操作结果是在图G中删除关键字v对应的顶点以及相关的弧；

（9）插入弧：函数名称是InsertArc(G,v,w)；初始条件是图G存在，v、w是和G中顶点关键字类型相同的给定值；操作结果是在图G中增加弧<v,w>，如果图G是无向图，还需要增加<w,v>；

（10）删除弧：函数名称是DeleteArc(G,v,w)；初始条件是图G存在，v、w是和G中顶点关键字类型相同的给定值；操作结果是在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>；

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

（12）广度优先搜索遍历：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。

附加功能：

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

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

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

（4）实现图的文件形式保存：其中，①需要设计文件数据记录格式，以高效保存图的数据逻辑结构(D,{R})的完整信息；②需要设计图文件保存和加载操作合理模式。附录B提供了文件存取的方法；

注：①保存到文件后，可以直接加载文件生成图。

（5）实现多个图管理：设计相应的数据结构管理多个图的查找、添加、移除等功能。

注：附加功能（5）实现多个图管理应能创建、添加、移除多个图，单图和多图的区别仅仅在于图管理的个数不同，多图管理应可自由切换到管理的每个图，并可单独对某图进行单图的所有操作。

## 4.3 实验任务

采用邻接表作为图的物理结构，实现4.2节的基本运算，可任选无向图、有向图、无向网和有向网这四种图中的一种实现。其中ElemType为数据元素的类型名，具体含义可自行定义，但要求顶点类型为结构型，至少包含二个部分，一个是能唯一标识一个顶点的关键字（类似于学号或职工号），另一个是其它部分。其它有关类型和常量的定义和引用详见文献[1]的p10。

构造一个具有菜单的功能演示系统。其中，在主程序中完成函数调用所需实参值的准备和函数执行结果的显示，并给出适当的操作提示显示（供参考）。

# 参考文献

[1] 严蔚敏等.数据结构（C语言版）.清华大学出版社

[2] [Larry Nyhoff](http://www.calvin.edu/~nyhl/index.html). [ADTs, Data Structures, and Problem Solving with C++.](http://vig.prenhall.com/catalog/academic/product/0,1144,0131409093,00.html)Second Edition,[Calvin College](http://cs.calvin.edu/),2005

[3] 殷立峰. Qt C++跨平台图形界面程序设计基础. 清华大学出版社,2014:192～197

[4] 严蔚敏等.数据结构题集（C语言版）.清华大学出版社

# 附录A顺序表实现框架程序清单

/\* Linear Table On Sequence Structure \*/

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

/\*---------page 10 on textbook ---------\*/

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

typedef int status;

typedefintElemType; //数据元素类型定义

/\*-------page 22 on textbook -------\*/

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

typedef struct{ //顺序表（顺序结构）的定义

ElemType \* elem;

int length;

intlistsize;

}SqList;

/\*-----page 19 on textbook ---------\*/

status InitList(SqList& L);

//status DestroyList(SqList& L);

//status ClearList(SqList&L);

//status ListEmpty(SqList L);

//intListLength(SqList L);

//status GetElem(SqListL,inti,ElemType& e);

//status LocateElem(SqListL,ElemType e); //简化过

//status PriorElem(SqListL,ElemTypecur,ElemType&pre\_e);

//status NextElem(SqListL,ElemTypecur,ElemType&next\_e);

//status ListInsert(SqList&L,inti,ElemType e);

status ListDelete(SqList&L,inti,ElemType& e);

status ListTrabverse(SqList L); //简化过

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

void main(void){

SqList L; int op=1;

while(op){

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

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

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

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

printf(" 2. DestroyList8. 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(" 0. Exit\n");

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

printf(" 请选择你的操作[0~12]:");

scanf("%d",&op);

switch(op){

case 1:

//printf("\n----IntiList功能待实现！\n");

if(InitList(L)==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:

printf("\n----DestroyList功能待实现！\n");

getchar();getchar();

break;

case 3:

printf("\n----ClearList功能待实现！\n");

getchar();getchar();

break;

case 4:

printf("\n----ListEmpty功能待实现！\n");

getchar();getchar();

break;

case 5:

printf("\n----ListLength功能待实现！\n");

getchar();getchar();

break;

case 6:

printf("\n----GetElem功能待实现！\n");

getchar();getchar();

break;

case 7:

printf("\n----LocateElem功能待实现！\n");

getchar();getchar();

break;

case 8:

printf("\n----PriorElem功能待实现！\n");

getchar();getchar();

break;

case 9:

printf("\n----NextElem功能待实现！\n");

getchar();getchar();

break;

case 10:

printf("\n----ListInsert功能待实现！\n");

getchar();getchar();

break;

case 11:

printf("\n----ListDelete功能待实现！\n");

getchar();getchar();

break;

case 12:

//printf("\n----ListTrabverse功能待实现！\n");

if(!ListTrabverse(L)) printf("线性表是空表！\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 InitList(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 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");

returnL.length;

}

# 附录B数据元素的文件读写

#include <stdio.h>

#include <stdlib.h>

typedef struct {

char c; int d; float f;

} ElemType;

typedef struct {

ElemType elem[10]; int length;

} SqList;

SqList L={{{'a',1,1.1},{'b',2,2.2},{'c',3,3.3},{'d',4,4.4}}, 4};

int main(intargc, char \*argv[]) {

FILE \*fp;char filename[30];inti;

printf("input file name: ");

scanf("%s",filename);

//写文件的方法

if ((fp=fopen(filename,"wb"))==NULL)

{

printf("File open erroe\n ");

return 1;

}

fwrite(L.elem,sizeof(ElemType),L.length,fp);

//这里是1次性写入，对于其它物理结构，可通过遍历，逐个访问数据元素

//并写入到文件中。也可以先写入表长，再写入全部元素，这样读入会更方便

fclose(fp);

//读文件的方法

L.length=0;

if ((fp=fopen(filename,"rb"))==NULL)

{

printf("File open erroe\n ");

return 1;

}

while(fread(&L.elem[L.length],sizeof(ElemType),1,fp))

L.length++;

//这里从文件中逐个读取数据元素恢复顺序表，对于不同的物理结构，可通过读//取的数据元素恢复内存中的物理结构。

fclose(fp);

return 0;

}

# 附录C线性表模板实例

#include "iostream"

using namespace std;

typedef float ElemType;

template <typename T>

struct list{

T elem[10];

int length;

};

template <typename T1>

int ListInsert(struct list<T1>&L,inti, T1 x)

{

int j;

if(i<1 || i>L.length+1) return ERROR;

if(i>100) return OVERFLOW;

for(j=L.length-1;j>=i-1;j--)

L.elem[j+1]=L.elem[j];

L.elem[i-1]=x;

L.length++;

return OK;

}

template <typename T2>

void ListDisplay(struct list<T2>&L)

{

inti;

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

cout<<" "<<L.elem[i];

}

int main()

{

struct list<ElemType> L1;

L1.length=0;

ListInsert(L1,1,(ElemType)10.2);

ListInsert(L1,1,(ElemType)20);

ListDisplay(L1);

return 1;

}