

**数 据 结 构 实 验**

**(2014级计算机科学与技术&信息安全专业适用)**

**计算机科学与技术学院**

**数组结构课程组**

**2015年8月**

目录

[1 基于顺序存储结构实现线性表的基本运算 1](#_Toc427723275)

[1.1 实验目的 1](#_Toc427723276)

[1.2 顺序表基本操作定义 1](#_Toc427723277)

[1.3 实验任务 2](#_Toc427723278)

[2 基于链式实现线性表的基本运算 3](#_Toc427723279)

[2.1 实验目的 3](#_Toc427723280)

[2.2 单链表基本操作定义 3](#_Toc427723281)

[2.3 实验任务 4](#_Toc427723282)

[3 基于顺序存储结构实现栈的基本运算 5](#_Toc427723283)

[3.1 实验目的 5](#_Toc427723284)

[3.2 栈基本操作定义 5](#_Toc427723285)

[3.3 实验任务 5](#_Toc427723286)

[4 基于循环队列存储结构实现队列的基本运算 6](#_Toc427723287)

[4.1 实验目的 6](#_Toc427723288)

[4.2 队列基本操作定义 6](#_Toc427723289)

[4.3 实验任务 6](#_Toc427723290)

[5 基于二叉链表实现二叉树的基本运算 7](#_Toc427723291)

[5.1 实验目的 7](#_Toc427723292)

[5.2 二叉树基本操作定义 7](#_Toc427723293)

[5.3 实验任务 9](#_Toc427723294)

[6 基于邻接表实现图的基本和常见运算 10](#_Toc427723295)

[6.1 实验目的 10](#_Toc427723296)

[6.2 图基本操作定义 10](#_Toc427723297)

[6.3 实验任务 11](#_Toc427723298)

[参考文献 12](#_Toc427723299)

# 1 基于顺序存储结构实现线性表的基本运算

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,熟练掌握线性表的基本运算的实现。

1.2 基于顺序存储结构的线性表基本操作描述

（1）InitaList(&L)

操作结果：构造一个空的线性表。

（2）DestroyList(&L)

初始条件：线性表L已存在。

操作结果：销毁线性表L。

（3）ClearList (&L)

初始条件：线性表L已存在。

操作结果：将L重置为空表。

（4）ListEmpty(L)

初始条件：线性表L已存在。

操作结果：若L为空表，则返回TRUE,否则返回FALSE。

（5）ListLength(L)

初始条件：线性表已存在。

操作结果：返回L中数据元素的个数。

（6）GetElem(L, i,&e)

初始条件：线性表已存在，1≤i≤ListLength(L)。

操作结果：用e返回L中第i个数据元素的值。

（7）LocateElem(L, e，compare（）)

初始条件：线性表已存在。

操作结果：返回L中第1个与e满足关系compare（）关系的数据元素的

位序，若这样的数据元素不存在，则返回值为0。

（8）PriorElem（L，cur\_e，&pre\_e）

初始条件：线性表L已存在。

操作结果：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）NextElem（L，cur\_e，&next\_e）

初始条件：线性表L已存在。

操作结果：若cur\_e是L的数据元素，且不是最后一个，则用next\_e返回它

的后继，否则操作失败，next\_e无定义。

（10）ListInsert(&L,i,e)

初始条件：线性表L已存在且非空，1≤i≤ListLength(L)+1。

操作结果：在L的第i个位置之前插入新的数据元素e，L的长度加1

（11）ListDelete(&L,i,&e)

初始条件：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，L的长度减1.

（12）ListTraverse(L，visit（）)

初始条件：线性表L已存在。

操作结果：依次对L的每个数据元素调用函数visit()。一旦调用失败，则操

作失败。

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的基本操作。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。要求构造成具有功能菜单的系统，线性表采用文件形式保存。考虑(选做)如何实现多个数据元素类型相同的线性表管理，可以参考图1-1(a)或图1-1(b)所示物理结构形式。

 

(a) (b)

图1-1 多顺序表的物理结构示意图

# 2 基于链式实现线性表的基本运算

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）物理结构采用带表头结点的单链表,熟练掌握线性表基本运算的实现。

2.2基于单链表存储结构的线性表基本操作描述

（1）InitaList(&L)

操作结果：构造一个空的单链表。

（2）DestroyList(&L)

初始条件：单链表L已存在。

操作结果：销毁单链表L。

（3）ClearList (&L)

初始条件：单链表L已存在。

操作结果：将L重置为空单链表。

（4）ListEmpty(L)

初始条件：单链表L已存在。

操作结果：若L为空单链表，则返回TRUE,否则返回FALSE.

（5）ListLength(L)

初始条件：单链表已存在。

操作结果：返回L中数据元素的个数。

（6）GetElem(L, i,&e)

初始条件：单链表已存在，1≤i≤ListLength(L)。

操作结果：用e返回L中第i个结点的数据元素值。

（7）LocateElem(L, e，compare（）)

初始条件：单链表已存在。

操作结果：返回L中第1个与e满足关系compare（）的数据元素结点的指

针，若这样的数据元素不存在，则返回值为NULL。

（8）PriorElem（L，cur\_e，&pre\_e）

初始条件：单链表L已存在。

操作结果：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）NextElem（L，cur\_e，&next\_e）

初始条件：单链表L已存在。

操作结果：若cur\_e是L的数据元素，且不是最后一个，则用next\_e返回它

的后继，否则操作失败，next\_e无定义。

（10）ListInsert(&L,i,e)

初始条件：单链表L已存在且非空，1≤i≤ListLength(L)+1。

操作结果：在L的第i个结点之前插入新数据元素e的结点。

（11）ListDelete(&L,i,&e)

初始条件：单链表L已存在且非空，1≤i≤ListLength(L)。

操作结果： 删除L第i个数据元素的结点，用e返回其结点数据元素的值。

（12）ListTraverse(L，visit（）)

初始条件：单链表L已存在。

操作结果：依次对L的每个数据元素调用函数visit()。一旦调用失败，则操

作失败。

2.3 实验任务

采用带表头结点的单链表作为线性表的物理结构，实现2.2的基本操作。要求构造成具有功能菜单的系统，实现对多个单链表实现操作，可参考图2-1所示物理结构。

图2-1 多线性表的物理结构示意图

# 3 基于顺序存储结构实现栈的基本运算

3.1 实验目的

通过实验达到：（1）加深对栈的概念、基本运算的理解；（2）熟练掌握栈的逻辑结构与物理结构的关系；（3）熟练掌握顺序栈的基本运算的实现；（4）通过栈的应用体会其用途。

3.2 栈基本操作定义

（1）InitStack(&S)

操作结果：构造一个空栈S。

（2）DestroyStack(&S)

初始条件：栈S存在。

操作结果：栈S被销毁，不在存在。

（3）ClearStack(&S)

初始条件：栈S存在。

操作结果：将S清成空栈。

（4）StackEmpty(S)

初始条件：栈S存在。

操作结果：若S为空栈,则返回值为TRUE；否则为FALSE。

（5）StackLength(S)

初始条件：栈S存在。

操作结果：返回栈S的元素个数。

（6）GetTop(S,&e)

初始条件：栈S存在并且非空。

操作结果：将栈顶元素拷贝到e。

（7）Push(&S,e)

初始条件：栈S存在。

操作结果：插入元素e为新的栈顶元素。

（8）Pop(&S,&e)

初始条件：栈S存在并且非空。

操作结果：删除栈S的栈顶元素，并送入e。

（9）StackTravserse(S,visit())

初始条件：栈S存在。

操作结果：从栈底到栈顶依次对栈S中的元素使用函数visit进行访问。

3.3 实验任务

1. 采用顺序栈作为栈的物理结构，实现3.2所列栈的全部操作。建议使用模板[3] 的方式实现。

2. 利用栈，实现表达式的求值，这里仅要求对含运算符+、-、\*、/和1位数的操作数的表达式求值。

# 4 基于循环队列存储结构实现队列的基本运算

4.1 实验目的

通过实验达到：（1）加深对队列的概念、基本运算的理解；（2）熟练掌握队列的逻辑结构与物理结构的关系；（3）熟练掌握循环队列的算法实现。

4.2 队列基本操作定义

（1）InitQueue(&Q)

操作结果：构造一个空队列Q。

（2） DestroyQueue(&Q)

初始条件：队列Q存在。

操作结果：将队列Q销毁，不再存在。

（3）ClearQueue(&Q)

初始条件：队列Q存在。

操作结果：将队列Q清为空队列。

（4）QueueEmpty(Q)

初始条件：队列Q存在。

操作结果：若队列Q为空队列，返回TRUE，否则返回FALSE。

（5）QueueLength(Q)

初始条件：队列Q存在。

操作结果：返回队列元素个数。

（6）GetHead(Q,&e)

初始条件：队列Q存在并且非空。

操作结果：读取队列Q的首元素,送e返回其值。

（7）EnQueue(&Q,e)

初始条件：队列Q存在。

操作结果：插入元素e到队列Q中作为尾元素。

（8）DeQueue(&Q,&e)

初始条件：队列Q存在并且非空。

操作结果：删除队列Q的首元素,并且用e返回其值。

（9）QueueTravserse(Q,visit())

初始条件：队列Q存在并且非空。

操作结果：从队首到队尾依次对队列中的元素使用函数visit进行访问。

4.3 实验任务

采用循环队列作为队列的物理结构，实现的队列的全部基本操作。建议使用模板的方式实现。

# 5 基于二叉链表实现二叉树的基本运算

5.1 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。

5.2 二叉树基本操作定义

（1）InitBiTree(&T)

操作结果：构造空二叉树T。

（2）DestroyBiTree(&T)

初始条件：二叉树T已存在。

操作结果：销毁二叉树T。

（3）CreateBiTree(&T,definition)

初始条件：definition 给出二叉树T的定义。

操作结果：按definition构造二叉树T。

（4）ClearBiTree (&T)

初始条件：二叉树T存在。

操作结果：将二叉树T清空。

（5）BiTreeEmpty(T)

初始条件：二叉树T存在。

操作结果：若T为空二叉树，则返回TRUE,否则返回FALSE.

（6）BiTreeDepth(T)

初始条件：二叉树T存在。

操作结果：返回T的深度。

（7）Root(T)

初始条件：二叉树T已存在。

操作结果：返回T的根。

（8）Value(T，e)

初始条件：二叉树T已存在，e是T中的某个结点。

操作结果：返回e的值。

（9）Assign(T，&e，value)

初始条件：二叉树T已存在，e是T中的某个结点。

操作结果：结点e赋值为value。

（10）Parent(T，e)

初始条件：二叉树T已存在，e是T中的某个结点。

操作结果：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。

（11）LeftChild(T，e)

初始条件：二叉树T存在，e是T中某个节点。

操作结果：返回e的左孩子结点指针。若e无左孩子，则返回NULL。

（12）RightChild(T，e)

初始条件：二叉树T已存在，e是T中某个结点。

操作结果：返回e的右孩子结点指针。若e无右孩子，则返回NULL。

（13）LeftSibling(T，e)

初始条件：二叉树T存在，e是T中某个结点。

操作结果：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。

（14）RightSibling(T，e)

初始条件：二叉树T已存在，e是T中某个结点。

操作结果：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。

（15）InsertChild(T,p,LR,c)

初始条件：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：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。

（16）DeleteChild(T，p，LR)

初始条件：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：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。

（17）PreOrderTraverse（T,Visit（））

初始条件：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。

（18）InOrderTraverse（T,Visit（））

初始条件：二叉树T存在，Visit是对结点操作的应用函数。

操作结果：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。

（19）PostOrderTraverse（T,Visit（））

初始条件：二叉树T存在，Visit是对结点操作的应用函数。

操作结果：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。

（20）LevelOrderTraverse（T,Visit（））

初始条件：二叉树T存在，Visit是对结点操作的应用函数。

操作结果：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。

5.3 实验任务

实现对多个结点元素类型相同的二叉树的管理。其中每一个二叉树采用二叉链表作为物理结构，每一个二叉树的属性至少包含有二叉树名称。采用顺序表的方式管理多个线性表。其物理结构如图5-1所示。



图5-1 多二叉树管理的物理结构示意图

要求构造成具有功能菜单的系统，能对每棵二叉树实现二叉树的基本操作；按自己的理解适当增加一些功能，如查找、统计等；多个二叉树采用1个或多个文件保存。

# 6 基于邻接表实现图的基本和常见运算

6.1 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）熟练掌握图的基本运算在不同的物理结构下的实现。

6.2 图基本操作定义

（1）CreateCraph(&G,V,VR)

初始条件：V是图的顶点集，VR是图的关系集。

操作结果：按V和VR的定义构造图G。

（2）DestroyCraph(&G)

初始条件：图G存在。

操作结果：销毁图G。

（3）LocateVex(G,u)

初始条件：图G存在，u和G中的顶点具有相同特征。

操作结果：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。

（4）FirstAdjVex(&G, v)

初始条件：图G存在，v是G的一个顶点。

操作结果：返回v的第一个邻接顶点，如果v没有邻接顶点，返回空。

（5）NextAdjVex(&G, v, w)

初始条件：图G存在，v是G的一个顶点,w是v的邻接顶点。

操作结果：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。

（6）InsertVex(&G,v)

初始条件：图G存在，v和G中的顶点具有相同特征。

操作结果：在图G中增加新顶点v。

（7）DeleteVex(&G,v)

初始条件：图G存在，v是G的一个顶点。

操作结果：在图G中删除顶点v和与v相关的弧。

（8）InsertArc(&G,v,w)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中增加弧<v,w>，如果图G是无向图，还需要增加<w,v>。

（9）DeleteArc(&G,v,w)

初始条件：图G存在，v、w是G的顶点。

操作结果：在图G中删除弧<v,w>，如果图G是无向图，还需要删除<w,v>。

（10）DFSTraverse(G,visit())

初始条件：图G存在。

操作结果：对图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。

（11）BFSTraverse(G,visit())

初始条件：图G存在。

操作结果：对图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。

6.3 实验任务

采用图的采用邻接矩阵表示法（Representation），实现无向（或有向）图（或网）的基本运算。构造成具有功能菜单的系统，采用文件形式保存图。

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