

**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：**

**姓 名：**

**指导教师：**

**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 2](#_Toc458159879)

[1.1 问题描述 2](#_Toc458159880)

[1.2 系统设计 2](#_Toc458159882)

[1.3 系统实现 2](#_Toc458159883)

[1.4 实验小结 2](#_Toc458159884)

[2 基于链式存储结构的线性表实现 2](#_Toc458159885)

[2.1 问题描述 2](#_Toc458159886)

[2.2 系统设计 2](#_Toc458159887)

[2.3 系统实现 2](#_Toc458159888)

[2.4 实验小结 2](#_Toc458159889)

[3 基于二叉链表的二叉树实现 2](#_Toc458159890)

[3.1 问题描述 2](#_Toc458159891)

[3.2 系统设计 2](#_Toc458159892)

[3.3 系统实现 2](#_Toc458159893)

[3.4 实验小结 2](#_Toc458159894)

[4 基于二叉链表的二叉树实现 2](#_Toc458159895)

[4.1 问题描述 2](#_Toc458159896)

[4.2 系统设计 2](#_Toc458159897)

[4.3 系统实现 2](#_Toc458159898)

[4.4 实验小结 2](#_Toc458159899)

[参考文献 2](#_Toc458159900)

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

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

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

[附录D 基于邻接表图实现的源程序 2](#_Toc458159904)

# 基于顺序存储结构的线性表实现

## 1.1问题描述

采用顺序表作为线性表的物理结构，实现线性表的基本运算：

1. 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。

2. 演示系统可选择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。

3. 顺序结构算法设计要求：

1) 初始化表：函数名称是InitiaList(&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。

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。

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()。

## 1.2 系统设计

1.2.1 系统总体设计：

通过switch函数进行功能的选取，然后调用相关功能的函数进行相关操作。

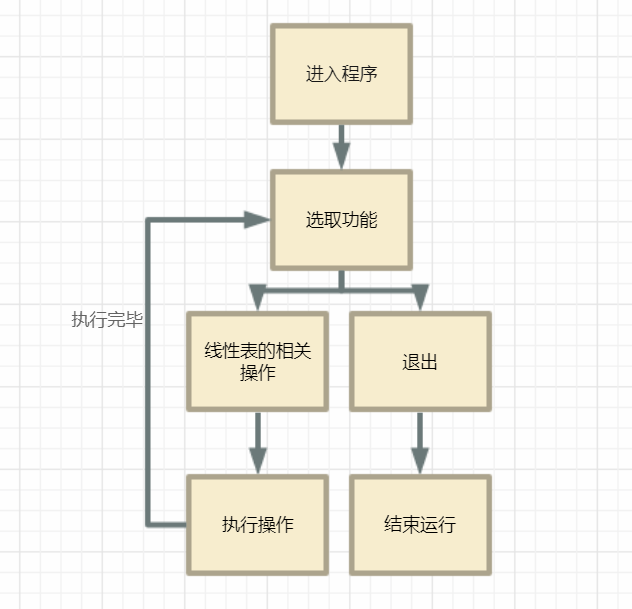


图1-1系统总体结构

1.2.2线性表物理结构

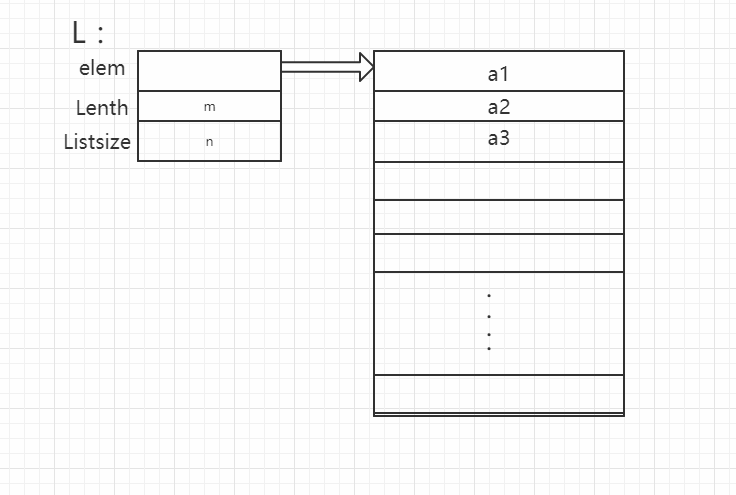


图1-2 线性表物理结构

1.2.3相关常量的类型与定义

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASTABLE -1

#define OVERFLOW -2

#define FILENAME "data"//数据保存的文件名

typedef int status; //函数返回状态

typedef int ElemType; //数据元素类型定义

#define LIST\_INIT\_SIZE 100//初始表大小

#define LISTINCREMENT 10//表增容大小

typedef struct{ //顺序表（顺序结构）的定义

ElemType \* elem;

int length;

int listsize;

}SqList;

1.2.4算法设计

* 1. InitiaList(&L)

算法思想：

1.申请存储数据的空间；

2.置表长为0。

操作结果：构造一个空的线性表

时间复杂度：O(1)

流程图：

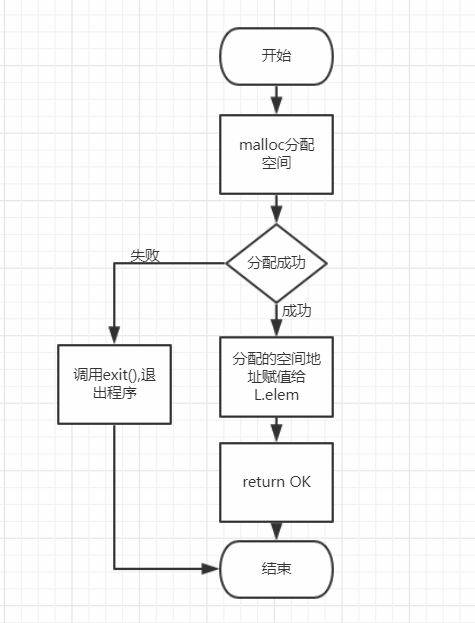


图1-3 InitiaList()流程图

* 1. DestroyList(&L)

算法思想：

1.释放线性表的存储空间；

2.置表指针为空

操作结果：销毁线性表L

时间复杂度：O(1)

流程图：

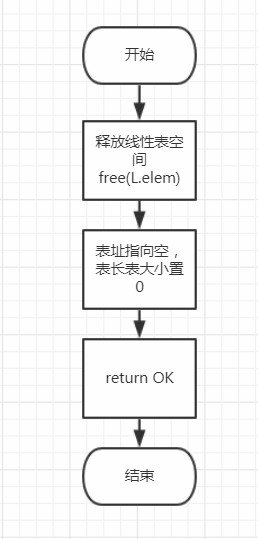


图1-4 DestroyList()流程图

* 1. ClearList(&L)

算法思想：置表长为0

操作结果：线性表L置空

时间复杂度：O(1)

流程图：

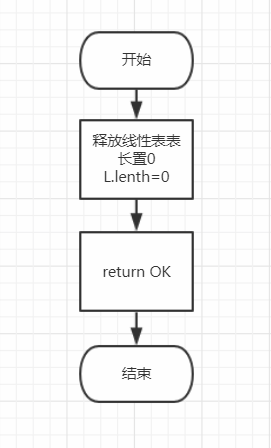


图1-5 ClearList()流程图

* 1. ListEmpty(L)

算法思想：若表长为0返回”TRUE”，否则返回”FALSE”。

操作结果：L为空返回TRUE,否则返回FALSE

时间复杂度：O(1)

流程图：

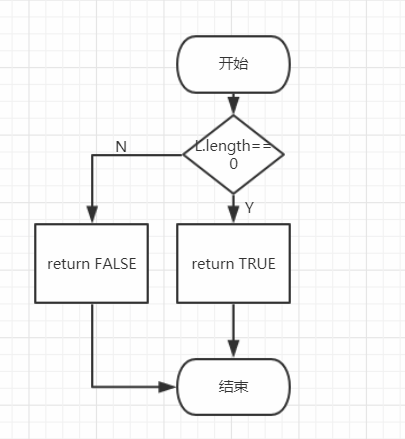


图1-6 ListEmpty()流程图

* 1. ListLength(L)

算法思想：返回表长

操作结果：返回线性表中元素的个数

时间复杂度：O(1)

流程图：

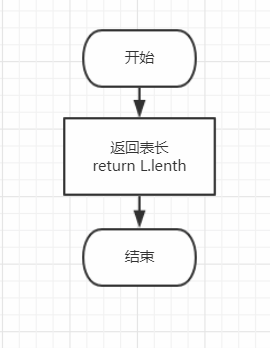


图1-7 ListLength(L)流程图

* 1. GetElem(L,i,&e)

算法思想：

1.寻址公式定位第i个元素。

2.将第i个元素赋值给e

操作结果：用e返回L中第i个数据元素的值

时间复杂度：O(1)

流程图：

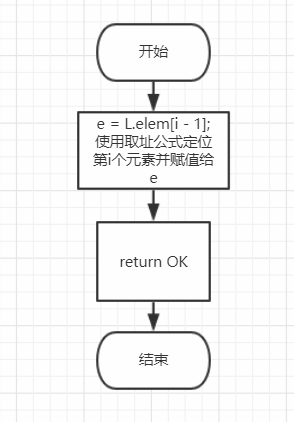


图1-8 GetElem()流程图

* 1. LocateElem(L,e,compare())

算法思想：

1.用compare()函数查找e;

2.找到返回TRUE,否则返回FALSE.

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。

时间复杂度：O(n)

流程图：

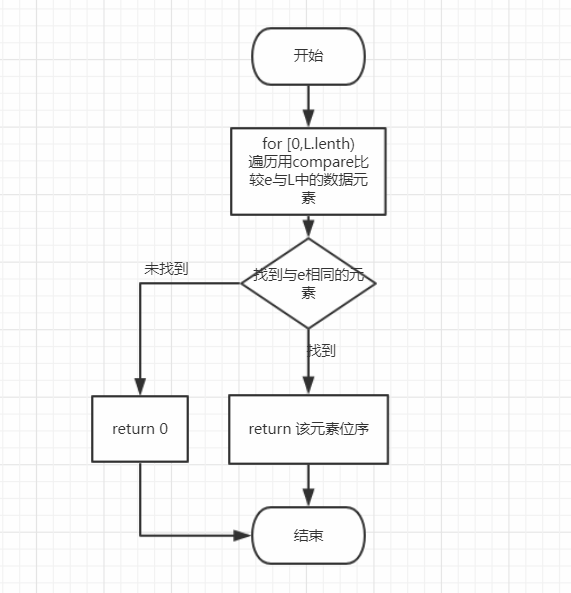


图1-9 LocateElem()流程图

* 1. PriorElem(L,cur\_e,&pre\_e)

算法思想：

1.查找cur\_e获得其序号order;

2.若order>1,将order-1单元的元素值赋值给pre\_e,否则返回FALSE。

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

时间复杂度：O(n)

流程图：

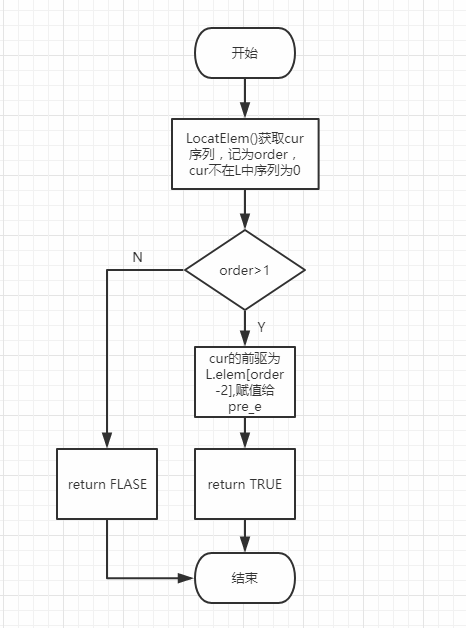


图1-10 PriorElem()流程图

* 1. NextElem(L,cur\_e,&next\_e)

算法思想：

1.查找cur\_e获得其序号order;

2.若order<表长,将order+1单元的元素值赋值next\_e,否则返回FALSE。

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

时间复杂度：O(n)

流程图：

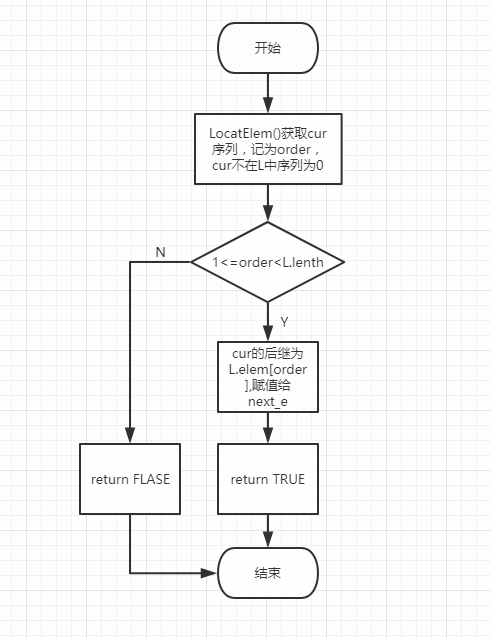


图1-11 NextElem()流程图

* 1. ListInsert(&L,i,e)

算法思想：

1.判断空间是否已满，若满则增配空间，并修改listsize

2.将序号为i-L.lenth的元素依次后移一位

3.位置i插入e

4.表长+1

操作结果：在L的第i个位置之前插入新的数据元素e。

时间复杂度：O(n)

流程图：

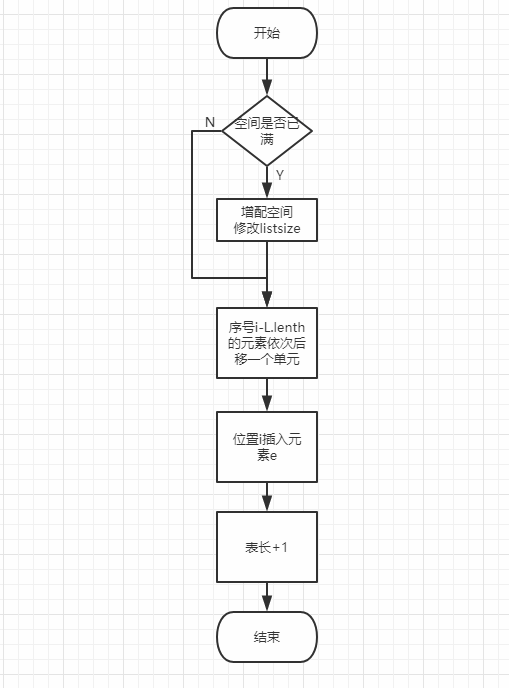


图1-12 ListInsert()流程图

* 1. ListDelete(&L,i,&e)

算法思想：

1. i单元的值赋值给e
2. 序号i+1-L.lenth的元素依次前移一个单元
3. 表长-1

操作结果：删除L的第i个数据元素，用e返回其值。

时间复杂度：O(n)

流程图：

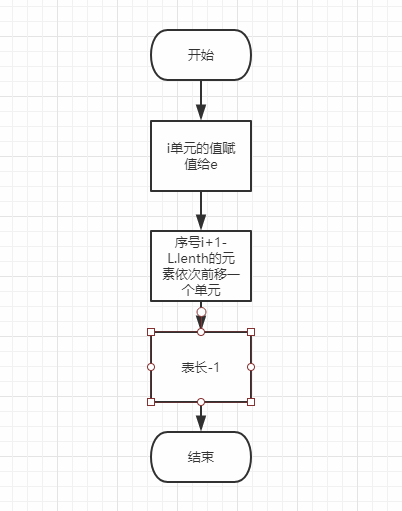


图1-13 ListDelete()流程图

* 1. ListTraverse(L,visit())

算法思想：使用visit()函数依次访问1-L.lenth数据元素

操作结果：对L的每个数据元素用函数visit()访问

时间复杂度：O(n)

流程图：

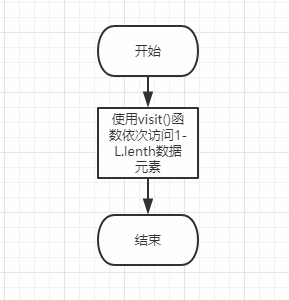


图1-14 ListTraverse()流程图

## 1.3 系统实现

××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××××× (如表1-1所示)

表1-1□××××××××××

|  |  |  |  |
| --- | --- | --- | --- |
| ××× | ××× | ××× | ××× |
| ××× | ×× | ×× | ×× |
| ××× | ×× | ×× | ×× |
| ××× | ×× | ×× | ×× |
| ××× | ×× | ×× | ×× |

.......

## 1.4 实验小结

# 2 基于链式存储结构的线性表实现

## 2.1 问题描述

## 2.2 系统设计

## 2.3 系统实现

## 2.4 实验小结

# 3 基于二叉链表的二叉树实现

## 3.1 问题描述

## 3.2 系统设计

## 3.3 系统实现

## 3.4 实验小结

# 4 基于邻接表的图实现

## 4.1 问题描述

## 4.2 系统设计

## 4.3 系统实现

## 4.4 实验小结

# 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

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

/\* 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

# 附录B 基于链式存储结构线性表实现的源程序

# 附录C 基于二叉链表二叉树实现的源程序

# 附录D 基于邻接表图实现的源程序