華中科技大學

课 程 实 验 报 告

课程名称： 数据结构实验

专业班级 CS 启明 2401

学　　号 U202414887

姓　　名 王李超

指导教师 王雄

报告日期 2025 年 5 月 1 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

1.1 问题描述

1.1.1 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。

1.1.2 具体问题

在设计线性表时，需要解决以下具体问题：

- **存储结构的选择**：选择顺序存储结构还是链式存储结构，需权衡存储效率和操作效率。
- **容量的动态扩展**：当线性表存储空间不足时，如何动态扩展存储容量以容纳更多元素。
- **基本操作的实现**：包括插入、删除、查找、更新等操作的具体实现及其时间复杂度优化。
- **边界条件处理**：如何处理空表、满表以及非法操作（如越界访问）等特殊情况。
- **数据类型的通用性**：设计线性表时，如何支持存储多种数据类型（如整数、浮点数、字符串等）。
- **内存管理**：如何高效管理内存，避免内存泄漏或冗余分配。
- **算法效率**：针对不同操作需求，优化算法以提高线性表的整体性能。

1.2 系统设计

整体系统结构设计方面，本系统采用模块化设计思想，将顺序表的各项操作功能（如初始化、插入、删除、查找、遍历、排序、文件读写等）分别封装为独立的函数，并通过主程序 main01.cpp 提供统一的菜单式交互界面，方便用户进行各类操作。系统支持多顺序表管理，用户可新建、删除、切换和重命名多个顺序表，提升了系统的灵活性和扩展性。各功能模块之间通过头文件（如 def.h、

func.h）进行数据类型和函数声明的解耦，便于维护和升级。

数据结构设计方面，核心采用顺序存储结构实现线性表。定义了 SqList 结构体，包含元素指针和当前长度等信息，实现了线性表的基本操作。为支持多顺序表管理，设计了 LISTS 结构体，内部维护一个顺序表数组，每个元素包含一个 SqList 及其名称。元素类型 ElemType 可根据实际需求灵活定义。通过结构体嵌套和指针管理，系统实现了对多个线性表的统一管理和操作，保证了数据的有序性和高效性。整体设计兼顾了功能完整性、易用性和可扩展性。

1.3 系统实现

主要说明各个主要函数的实现思想，复杂函数可辅助流程图进行说明，函数和系统实现的源代码放在附录中。

1.3.1 主要函数实现思想

- **InitList**：判断线性表是否已存在，若不存在则分配初始空间，初始化长度和容量。
- **DestroyList**：释放线性表空间，并将指针和长度等信息重置，防止内存泄漏。
- **ClearList**：不释放空间，仅将长度归零，实现逻辑清空。
- **ListEmpty**：判断线性表是否存在及是否为空，返回相应状态。
- **ListLength**：返回线性表当前长度，若不存在则返回异常。
- **GetElem**：获取指定位置元素，先判断合法性和存在性。
- **LocateElem**：顺序查找指定元素，返回其逻辑序号，未找到返回 0。
- **PriorElem/NextElem**：查找指定元素的前驱或后继，遍历查找并返回相邻元素。
- **ListInsert**：判断插入位置合法性，必要时扩容，移动元素后插入新元素。
- **ListDelete**：判断删除位置合法性，保存被删元素，移动后续元素覆盖。
- **ListTraverse**：顺序输出所有元素，便于调试和展示。
- **MaxSubArray**：实现最大连续子数组和的求解，采用动态规划思想。
- **SubArrayNum**：统计和为指定值的子数组个数，双重循环遍历所有子区间。
- **sortList**：采用冒泡排序对顺序表元素排序。
- **saveListToFile/loadListFromFile**：实现顺序表的文件保存与加载，便于数据持久化。

- **manageMultipleLists**：输出当前所有顺序表及其名称，实现多表管理。
- **AddList/RemoveList/LocateList**：实现多顺序表的添加、删除和查找，支持名称唯一性和内存管理。

1.3.2 部分函数实现方法

以下选取几个实现较为复杂或具有代表性的函数，简要说明其实现思路，并给出伪代码辅助理解。

1. ListInsert（顺序表插入） 该函数需判断插入位置是否合法，若空间不足则动态扩容，然后将插入位置及其后的元素依次后移，最后插入新元素。

算法 1.1. ListInsert（顺序表插入）

Input: 顺序表 L，插入位置 i，元素 e

Output: 插入是否成功

if L 不存在 **then**

return INFEASIBLE

end if

检查内存分配是否成功

for $j = L.length$ **downto** i **do**

$L.elem[j] \leftarrow L.elem[j-1]$

end for

将元素插入到位置 i：$L.elem[i-1] \leftarrow e$

return OK

2. ListDelete（顺序表删除） 首先判断删除位置是否合法，保存被删元素，然后将其后的元素依次前移，最后长度减一。

算法 1.2. ListDelete（顺序表删除）

Input: 顺序表 L，删除位置 i

Output: 被删除元素 e，删除是否成功

if L 不存在 **then**

return INFEASIBLE
end if
检查删除位置合法性
$e \leftarrow L.elem[i-1]$
for $j = i$ **to** $L.length - 1$ **do**
$L.elem[j-1] \leftarrow L.elem[j]$
end for
顺序表长度减一
return OK

3. MaxSubArray（最大连续子数组和） 采用动态规划思想，遍历数组，记录当前子数组和与最大值。

算法 1.3. MaxSubArray（最大连续子数组和）
Input: 顺序表 L
Output: 最大连续子数组和 $maxSum$
$maxSum \leftarrow L.elem[0]$
$currentSum \leftarrow 0$
for $i = 0$ **to** $L.length - 1$ **do**
if $currentSum > 0$ **then**
$currentSum \leftarrow currentSum + L.elem[i]$
else
$currentSum \leftarrow L.elem[i]$
end if
if $currentSum > maxSum$ **then**
$maxSum \leftarrow currentSum$
end if
end for
return $maxSum$

4. AddList（多顺序表添加） 支持批量添加，需判断名称唯一性，动态分配空间，并循环插入元素直到输入 0 结束。

算法 1.4. AddList（多顺序表添加）

Input: 多顺序表 $Lists$，新表名称 $ListName$

Output: 添加是否成功

```
for 每个待添加顺序表 do
    while 名称重复 do
        提示重新输入 ListName
    end while
    if 顺序表数量已达上限 then
        return ERROR
    end if
    分配新表空间，初始化
    while 输入元素 e ≠ 0 do
        插入 e 到新表
    end while
    添加成功
end for
return OK
```

1.4 系统测试

主要说明针对各个函数正常和异常的测试用例及测试结果。测试用例设计时，考虑了正常情况、边界条件（如1-1中多次创建、删除线性表，查询边界处元素）和异常情况等多种场景，确保系统的健壮性和稳定性。以下是部分测试用例及其结果：

图 1-1　线性表操作界面

1.4.1　基础功能测试

表 1-1　线性表主要功能测试用例及结果

测试功能及序号	输入	输出
1. 构造空线性表	\	线性表创建成功!
1. 构造空线性表	\	线性表创建失败!
10. 插入元素（3 次）	4 1；6 2；8 3	插入成功!（3 次）
4. 判空线性表	\	线性表不是空表!
5. 求表长	\	线性表的长度为：3
6. 获取元素	2	线性表的第 2 个元素为：6
7. 定位元素	8	线性表中元素 8 的序号为：3
8. 获取前驱	4	线性表中元素 4 的前驱元素查找失败!
8. 获取前驱	6	线性表中元素 6 的前驱元素为：4
9. 获取后继	8	线性表中元素 8 的后继元素查找失败!

9. 获取后继	4	线性表中元素 4 的后继元素为：6
12. 遍历线性表	\	4 6 8
16. 文件保存/17. 文件读取	\	保存成功!/载入成功!
11. 删除元素	2	线性表中元素 6 删除成功！
3. 清空线性表	\	线性表清空成功!
2. 销毁线性表	\	线性表销毁成功!
2. 销毁线性表	\	线性表销毁失败!
0. 退出系统	\	欢迎再次使用本系统！

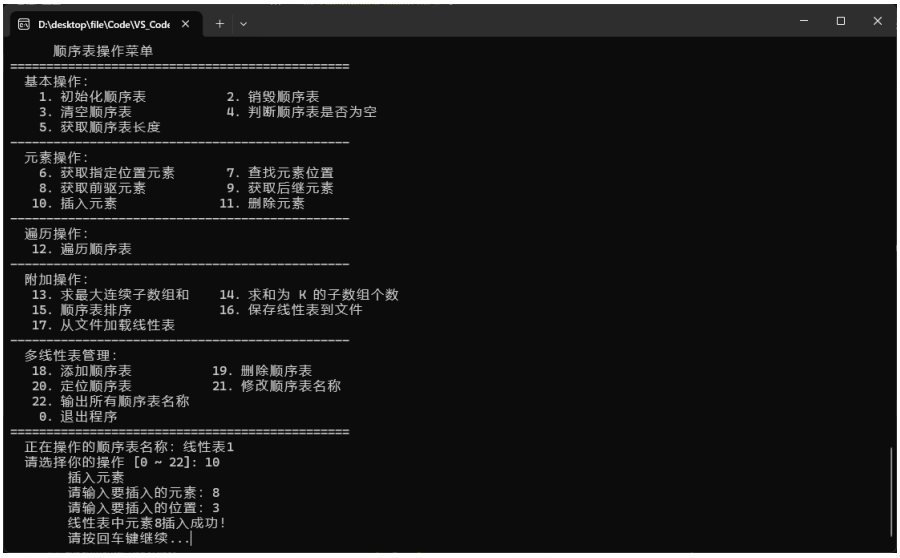

图 1-2　基础功能测试截图

1.4.2　附加功能测试

表 1-3　测试时初始状态

表名称	数据
a	无

b	20 1 2 -4 10 -19 2 2 50

表 1-5　线性表附加功能测试用例及结果

测试功能及序号	输入	输出
13. 最大连续子数组和	b	最大连续子数组和为：55
14. 和为指定值的子数组个数	1 2 -3 4 -5 6 7 -8 9 10	和为指定值的子数组个数为：0
15. 顺序表排序	b	排序成功！
12. 遍历线性表	\	20 1 2 -4 10 -19 2 2 50
16. 文件保存	\	保存成功!
3. 清空线性表	\	线性表清空成功!
17. 文件读取	\	载入成功!
12. 遍历线性表	\	20 1 2 -4 10 -19 2 2 50
22. 输出所有顺序表名称	\	线性表名称：a b

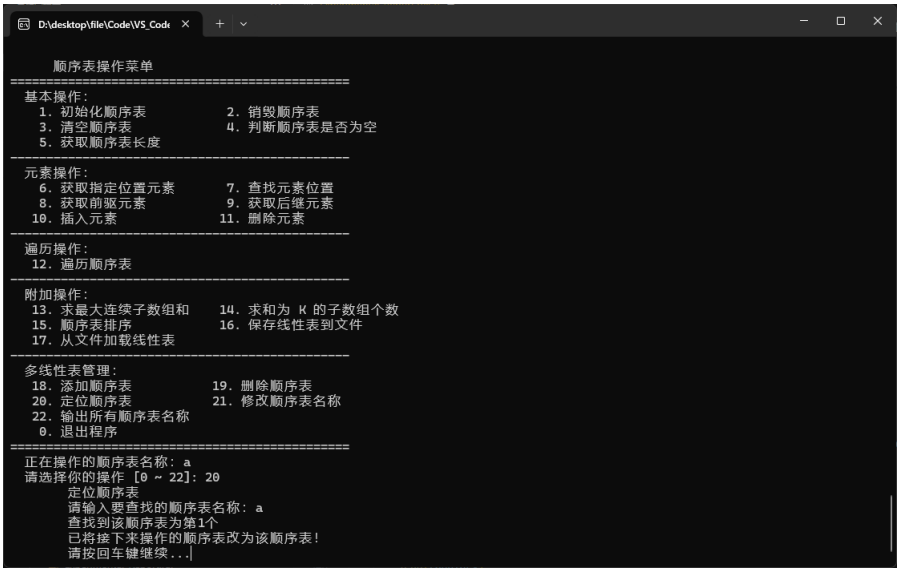

图 1-3　附加功能测试截图

1.4.3　多线性表管理功能测试

表 1-7　多线性表管理功能测试用例及结果

测试功能及序号	输入	输出
18. 添加线性表	2，a，b，20 1 2 -4 10 -19 2 2 50 0	添加成功！（2 次）
19. 删除线性表	线性表 2	删除成功！
20. 定位线性表	线性表 1	查找成功！
22.　输出所有顺序表名称	\	a, b
5. 重命名线性表	a，a1	重命名成功！
22.　输出所有顺序表名称	\	a1, b
0. 退出系统	\	欢迎再次使用本系统！

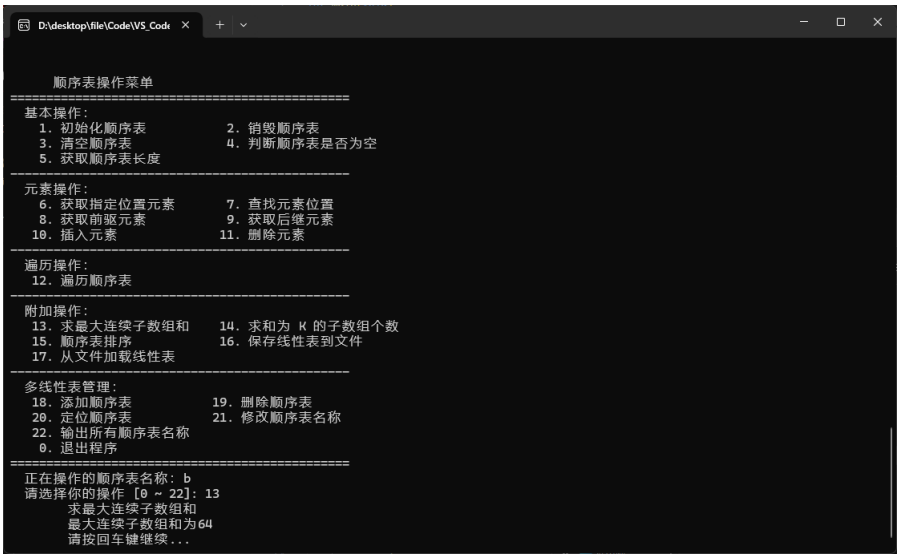

图 1-4　多线性表管理功能测试截图

1.5　实验小结

本次实验通过对线性表的实现与操作，深入理解了线性表的基本概念、存储结构及其基本运算。通过对顺序存储结构的实现，掌握了线性表的动态扩展、插入、删除、查找等操作的具体实现方法。同时，通过多顺序表管理功能，提升了

系统的灵活性和扩展性。实验中遇到的问题主要集中在内存管理和边界条件处理上，通过不断调试和优化，最终实现了一个功能完整、易用性强的线性表系统。

在设计线性表的过程中，遭遇了一些困难，如如何高效地管理内存、如何处理特殊情况（如空表、满表等）。通过查阅资料和反复测试，逐步解决了这些问题。此外，实验还涉及了文件读写操作的实现，增强了数据的持久化能力。通过对线性表的遍历和排序等操作，提升了系统的实用性和用户体验。整体而言，本次实验不仅加深了对线性表的理解，也提高了编程能力和问题解决能力。

此外，实验还涉及了文件读写操作的实现，增强了数据的持久化能力。通过对线性表的遍历和排序等操作，提升了系统的实用性和用户体验。整体而言，本次实验不仅加深了对线性表的理解，也提高了编程能力和问题解决能力。

2 基于邻接表的图实现

2.1 问题描述

在设计线性表时[1]

2.2 系统设计

2.3 系统实现

主要说明各个主要函数的实现思想，复杂函数可辅助流程图进行说明，函数和系统实现的源代码放在附录中。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。

2.4 系统测试

主要说明针对各个函数正常和异常的测试用例及测试结果画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。画图说明网页的整体框架，进行简要的文字描述等。

2.5 实验小结

3 课程的收获和建议

描述通过学习该专题，有何收获，有何建议，如某专题可适当减少讲授时间、某专题可适当增加讲授内容和时间等。描述通过学习该专题，有何收获，有何建议，如某专题可适当减少讲授时间、某专题可适当增加讲授内容和时间等。描述通过学习该专题，有何收获，有何建议，如某专题可适当减少讲授时间、某专题可适当增加讲授内容和时间等。描述通过学习该专题，有何收获，有何建议，如某专题可适当减少讲授时间、某专题可适当增加讲授内容和时间等。

3.1 基于顺序存储结构的线性表实现

描述通过学习计算机基础知识专题，有何收获，有何建议，如某专题可适当减少讲授时间、某专题可适当增加讲授内容和时间等。描述网页的设计和实现过程中遇到的问题及如何解决。描述网页的设计和实现过程中遇到的问题及如何解决。描述网页的设计和实现过程中遇到的问题及如何解决。描述网页的设计和实现过程中遇到的问题及如何解决。描述网页的设计和实现过程中遇到的问题及如何解决。描述网页的设计和实现过程中遇到的问题及如何解决。描述网页的设计和实现过程中遇到的问题及如何解决。描述网页的设计和实现过程中遇到的问题及如何解决。

3.2 基于二叉链表的二叉树实现

描述通过学习计算机基础知识专题，有何收获，有何建议，如某专题可适当减少讲授时间、某专题可适当增加讲授内容和时间等。描述通过学习计算机基础知识专题，有何收获，有何建议，如某专题可适当减少讲授时间、某专题可适当增加讲授内容和时间等。

参考文献

[1] BAFNA V, PEVZNER P A. Genome Rearrangements and Sorting by Reversals[J/OL]. SIAM J. Comput., 1996, 25(2): 272 – 289 [1996-02-01]. http://dx.doi.org/10.1137/S0097539793250627.

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 D 基于邻接表图实现的源程序