

东莞城市学院人工智能学院

School of Artificial Intelligence

《数据结构》

实验报告

| 学号 | 202435710252 |
| --- | --- |
| 姓名 | 钟润柯 |
| 专业、班级 | 2024级计科（专升本）A班 |
| 课程名称 | 《数据结构》 |
| 指导教师 | 李言一 |
| 学期 | 2024—2025第1学期 |
| 实验次序 | 实验一 |
| 上机时间 | 2024-9-25 周三 第5-6节 |
| 上机地点 | 3B416 |
| 分数 |  |

二○二三——二○二四学年 第一学期

实验一 顺序表的定义和运算

# 实验课时

课内：2课时

# 实验类型

操作性实验

# 实验目的

1.掌握线性表的逻辑特征。

2.掌握线性表顺序存储结构的特点

3.熟练掌握顺序表的基本运算。

4.能编程实现顺序线性表的创建、插入和删除。

5.逐步培养解决实际问题的编程能力。

# 实验内容和步骤

## 一、源代码

（1）顺序表结构的定义及初始化顺序表的源码

定义部分、函数部分和主函数生效部分：

|  |
| --- |
| 1. #include <stdio.h> 2. #define MaxSize 50 3. typedef int ElemType; 4. typedef struct SqList 5. { 6. ElemType data[MaxSize]; 7. int length 8. } SqList; 9. // 初始化顺序表 10. void InitList(SqList \*L) 11. { 12. L->length = 0; 13. } 14. // 打印顺序表 15. void PrintList(SqList \*L) 16. { 17. for (int i = 0; i < L->length; i++) 18. { 19. printf("%d ", L->data[i]); 20. } 21. printf("\n"); 22. } 23. int main() 24. { 25. SqList L; 26. printf("创建顺序表L\n"); 27. InitList(&L); 28. printf("初始化成功\n"); 29. // 模拟键盘输入数据 30. int elements[] = {21, 18, 30, 75, 42, 56}; 31. int n = sizeof(elements) / sizeof(elements[0]); 32. for (int i = 0; i < n; i++) 33. { 34. InsertElement(&L, i + 1, elements[i]); 35. } 36. printf("顺序表的元素有："); 37. PrintList(&L); 38. if (InsertElement(&L, 3, 67)) 39. { 40. printf("在第3个位置插入67成功\n"); 41. } 42. else 43. { 44. printf("在第3个位置插入67失败\n"); 45. } 46. if (InsertElement(&L, 9, 10)) 47. { 48. printf("在第9个位置插入10成功\n"); 49. } 50. else 51. { 52. printf("在第9个位置插入10失败\n"); 53. } 54. printf("插入后的顺序表为："); 55. PrintList(&L); 56. if (DeleteElement(&L, 6)) 57. { 58. printf("删除第6个元素成功\n"); 59. } 60. else 61. { 62. printf("删除第6个元素失败\n"); 63. } 64. if (DeleteElement(&L, 8)) 65. { 66. printf("删除第8个元素成功\n"); 67. } 68. else 69. { 70. printf("删除第8个元素失败\n"); 71. } 72. printf("删除后的顺序表中的元素："); 73. PrintList(&L); 74. return 0; 75. } |

（2）顺序表插入数据的源码：

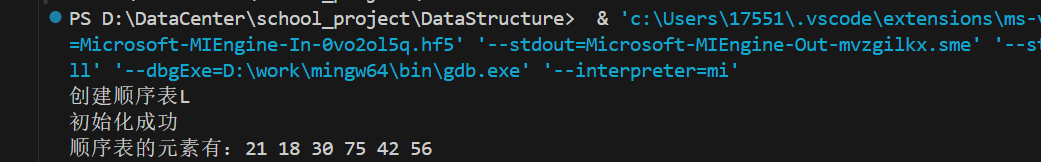
|  |
| --- |
| 1. // 插入元素 2. int InsertElement(SqList \*L, int index, ElemType element) 3. { 4. // 校验 5. if (L->length >= MaxSize) 6. { 7. printf("顺序表已满，无法插入\n"); 8. return 0; 9. } 10. if (index < 1 || index > L->length + 1) 11. { 12. printf("插入位置不合法\n"); 13. return 0; 14. } 15. // 循环插入数据 16. for (int i = L->length - 1; i >= index - 1; i--) 17. { 18. L->data[i + 1] = L->data[i]; 19. } 20. L->data[index - 1] = element; 21. L->length++; 22. return 1; 23. } |

（3）顺序表删除数据的源码：

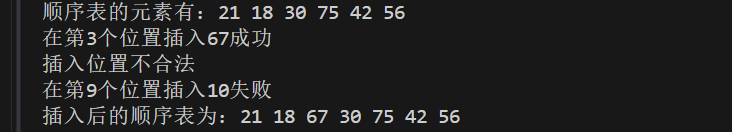
|  |
| --- |
| 1. // 删除元素 2. int DeleteElement(SqList \*L, int index) 3. { 4. if (index < 1 || index > L->length) 5. { 6. printf("删除位置不合法\n"); 7. return 0; 8. } 9. for (int i = index; i < L->length; i++) 10. { 11. L->data[i - 1] = L->data[i]; 12. } 13. L->length--; 14. return 1; 15. } |

## }二、程序运行结果截图

1. 顺序表结构的定义及初始化顺序表的截图：



1. 顺序表插入数据的截图：



1. 顺序表删除数据的截图：



## 三、实验总结 顺序表是用一段连续的存储单元依次存储数据元素的线性结构。在C语言中，可以使用数组来实现顺序表。顺序表的操作包括初始化、插入、删除和打印等。初始化操作将顺序表的长度设置为0。插入操作在指定位置插入元素，需要移动后续元素以腾出空间。删除操作则删除指定位置的元素，并移动后续元素以填补空缺。打印操作用于输出顺序表中的所有元素。顺序表的特点是随机访问效率高，但插入和删除操作可能需要移动大量元素，效率较低。在实际应用中，顺序表适用于元素数量变化不大且频繁进行随机访问的场景。