计算机科学技术学院实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **课程名称** | 面向对象程序设计 | | | **学 号** |  |
| **实验项目** | 实验一 | | | **姓 名** |  |
| **学 时** | 2 | **项目性质** | 综合型 | **班 级** | 2405111B |
| **指导教师** | 蒋振刚 | **实验地点** | 实训楼424 | **日 期** | 2025年3月24日 |
| 1. **实验目的和要求**   用数组实现线性表。  目的：练习类的设计和使用。体会抽象数据类型的内涵。  要求：设计和实现一个线性表类SList，满足：  1）用数组实现存储；  2）实现线性表的基本操作；  3）测试代码验证 List的基本操作。 | | | | | |
| 1. **实验环境**   系统：fedora 41  环境：g++ | | | | | |
| 1. **实验内容与过程**   **SList.h:**  **#ifndef SLIST\_H**  **#define SLIST\_H**  **#include <iostream>**  **// 定义最大容量**  **const int MAX\_SIZE = 100;**  **// 线性表类定义**  **class SList {**  **public:**  **// 构造函数**  **SList();**    **// 析构函数**  **~SList();**    **// 清空线性表**  **void clear();**    **// 判断线性表是否为空**  **bool isEmpty() const;**    **// 获取线性表长度**  **int size() const;**    **// 获取指定位置的元素**  **bool getElem(int i, int &e) const;**    **// 查找元素位置（返回第一次出现的位置，未找到返回0）**  **int locate(int e) const;**    **// 在指定位置插入元素**  **bool insert(int i, int e);**    **// 删除指定位置的元素**  **bool remove(int i, int &e);**    **// 打印线性表**  **void display() const;**    **private:**  **int data[MAX\_SIZE]; // 数据数组**  **int length; // 当前长度**  **};**  **// 构造函数**  **SList::SList() : length(0) {}**  **// 析构函数**  **SList::~SList() {}**  **// 清空线性表**  **void SList::clear() {**  **length = 0;**  **}**  **// 判断线性表是否为空**  **bool SList::isEmpty() const {**  **return length == 0;**  **}**  **// 获取线性表长度**  **int SList::size() const {**  **return length;**  **}**  **// 获取指定位置的元素**  **bool SList::getElem(int i, int &e) const {**  **// 检查位置是否有效（i从1开始计数）**  **if (i < 1 || i > length) {**  **return false;**  **}**  **e = data[i-1]; // 数组从0开始，因此要减1**  **return true;**  **}**  **// 查找元素位置（返回第一次出现的位置，未找到返回0）**  **int SList::locate(int e) const {**  **for (int i = 0; i < length; i++) {**  **if (data[i] == e) {**  **return i + 1; // 返回逻辑位置（从1开始）**  **}**  **}**  **return 0; // 未找到**  **}**  **// 在指定位置插入元素**  **bool SList::insert(int i, int e) {**  **// 检查线性表是否已满**  **if (length >= MAX\_SIZE) {**  **std::cout << "线性表已满，无法插入！" << std::endl;**  **return false;**  **}**    **// 检查插入位置是否有效（i可以从1到length+1）**  **if (i < 1 || i > length + 1) {**  **std::cout << "插入位置无效！" << std::endl;**  **return false;**  **}**    **// 将第i个位置及之后的元素后移**  **for (int j = length; j >= i; j--) {**  **data[j] = data[j-1];**  **}**    **// 插入元素**  **data[i-1] = e;**  **length++;**  **return true;**  **}**  **// 删除指定位置的元素**  **bool SList::remove(int i, int &e) {**  **// 检查位置是否有效**  **if (i < 1 || i > length) {**  **std::cout << "删除位置无效！" << std::endl;**  **return false;**  **}**    **// 保存被删除的元素**  **e = data[i-1];**    **// 将第i个位置之后的元素前移**  **for (int j = i; j < length; j++) {**  **data[j-1] = data[j];**  **}**    **length--;**  **return true;**  **}**  **// 打印线性表**  **void SList::display() const {**  **if (isEmpty()) {**  **std::cout << "线性表为空！" << std::endl;**  **return;**  **}**    **std::cout << "线性表内容：";**  **for (int i = 0; i < length; i++) {**  **std::cout << data[i] << " ";**  **}**  **std::cout << std::endl;**  **}**  **#endif**  **Main.cpp**  **#include <iostream>**  **#include "include/SList.h"**  **// 测试程序**  **int main() {**  **SList list;**  **int e;**    **std::cout << "测试线性表基本操作：" << std::endl;**    **// 测试插入操作**  **std::cout << "\n===== 测试插入操作 =====" << std::endl;**  **list.insert(1, 10);**  **list.insert(2, 20);**  **list.insert(3, 30);**  **list.insert(2, 15); // 在中间插入**  **list.display();**    **// 测试获取元素**  **std::cout << "\n===== 测试获取元素 =====" << std::endl;**  **if (list.getElem(2, e)) {**  **std::cout << "第2个元素是：" << e << std::endl;**  **}**    **// 测试查找元素**  **std::cout << "\n===== 测试查找元素 =====" << std::endl;**  **int pos = list.locate(20);**  **if (pos) {**  **std::cout << "元素20的位置是：" << pos << std::endl;**  **} else {**  **std::cout << "未找到元素20" << std::endl;**  **}**    **// 测试删除操作**  **std::cout << "\n===== 测试删除操作 =====" << std::endl;**  **if (list.remove(2, e)) {**  **std::cout << "删除的元素是：" << e << std::endl;**  **}**  **list.display();**    **// 测试其他操作**  **std::cout << "\n===== 测试其他操作 =====" << std::endl;**  **std::cout << "线性表长度：" << list.size() << std::endl;**  **std::cout << "线性表是否为空：" << (list.isEmpty() ? "是" : "否") << std::endl;**    **// 测试清空操作**  **std::cout << "\n===== 测试清空操作 =====" << std::endl;**  **list.clear();**  **std::cout << "清空后，线性表长度：" << list.size() << std::endl;**  **std::cout << "清空后，线性表是否为空：" << (list.isEmpty() ? "是" : "否") << std::endl;**  **list.display();**    **return 0;**  **}**  wps | | | | | |
| 1. **实验结果与分析**     分析：代码构建了 slist 类，其包含私有成员 length 与 data 数组，前者记录表长，后者存储元素。在类了定义获取长度、判断空满、插入、删除、查找元素位置及获取指定位置元素等函数。进行插入和删除操作进行时先进行边界检查，避免非法操作。采用遍历数组的方式进行查找操作在对类的操作。在主函数中先声明一个类，再进行各种操作进行测试，结束时自动调用析构函数清理数据。 | | | | | |
| 1. **实验心得**   在本次实验中，我设计并实现了一个基于数组的线性表类 SList，并完成了线性表的基本操作，包括插入、删除、查找等功能。通过这次实验，我对线性表这一数据结构有了更深的理解，同时也进一步熟悉了C++的类设计与使用。  1. 对线性表的理解  线性表是一种最基本的数据结构之一，它具有顺序存储和逻辑连续的特点。相比链表，顺序存储的线性表在随机访问时效率较高，但插入和删除操作需要移动元素，效率较低。本实验采用数组来存储线性表的数据，这意味着：  访问元素的时间复杂度是 O(1)，因为数组支持直接索引访问。  插入和删除的时间复杂度最差情况下是 O(n)，因为可能需要移动大量元素。  2. C++ 类的设计与实现  本次实验采用 C++ 类的方式封装线性表，主要涉及：  构造函数与析构函数：构造函数 SList() 负责初始化 length，析构函数 ~SList() 为空，因为 data 是静态数组，不需要手动释放内存。  成员函数的封装：所有操作均以成员函数的形式提供，保证了数据的封装性，符合面向对象编程的思想。  常量成员函数：如 isEmpty()、size() 和 getElem() 等函数不修改对象状态，因此定义为 const，提高了代码的安全性和可读性。 | | | | | |
| 1. **教师评语** | | | | | |
| 1. **实验成绩**   教师签名： 蒋振刚 批阅日期： 2025 年 3 月 24日 | | | | | |

注：项目性质为 演示型、验证型、设计型、综合型和创新型。