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** | 实验五 | | | **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** | 2024年4月21日 |
| 1. **实验目的和要求**   第5个实验：通过继承实现栈。  目的：练习类的继承（派生）。  要求：以DList为基类，派生栈类：DStack：  1）实现栈的基本操作；  2）编写测试代码验证栈的功能。 | | | | | |
| 1. **实验环境**   系统：fedora 42  环境：g++ | | | | | |
| 1. **实验内容与过程**   **第二次实验的DList.h:**  **#ifndef DLIST\_H**  **#define DLIST\_H**  **#include <iostream>**  **class DList {**  **public:**  **// 构造函数**  **DList();**    **// 析构函数**  **~DList();**    **// 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;**  **protected:**  **struct Node {**  **int data;**  **Node \*prev;**  **Node \*next;**  **};**  **Node \*head;**  **Node \*tail;**  **int length;**  **};**  **DList::DList() : length(0) {**  **head = new Node;**  **tail = new Node;**  **head->next = tail;**  **tail->prev = head;**  **}**  **DList::~DList() {**  **clear();**  **delete head;**  **delete tail;**  **}**  **void DList::clear() {**  **Node \*p = head->next;**  **while (p != tail) {**  **Node \*q = p;**  **p = p->next;**  **delete q;**  **}**  **head->next = tail;**  **tail->prev = head;**  **length = 0;**  **}**  **bool DList::isEmpty() const {**  **return length == 0;**  **}**  **int DList::size() const {**  **return length;**  **}**  **bool DList::getElem(int i, int &e) const {**  **if (i < 1 || i > length) {**  **std::cout << "位置无效！" << std::endl;**  **return false;**  **}**  **Node \*p = head->next;**  **for (int j = 1; j < i; j++) {**  **p = p->next;**  **}**  **e = p->data;**  **return true;**  **}**  **int DList::locate(int e) const {**  **Node \*p = head->next;**  **for (int i = 1; p != tail; i++) {**  **if (p->data == e) {**  **return i;**  **}**  **p = p->next;**  **}**  **return 0;**  **}**  **bool DList::insert(int i, int e) {**  **if (length >= 100) {**  **std::cout << "线性表已满，无法插入！" << std::endl;**  **return false;**  **}**  **if (i < 1 || i > length + 1) {**  **std::cout << "插入位置无效！" << std::endl;**  **return false;**  **}**  **Node \*p = head;**  **for (int j = 0; j < i; j++) {**  **p = p->next;**  **}**  **Node \*q = new Node;**  **q->data = e;**  **q->prev = p->prev;**  **q->next = p;**  **p->prev->next = q;**  **p->prev = q;**  **length++;**  **return true;**  **}**  **bool DList::remove(int i, int &e) {**  **if (i < 1 || i > length) {**  **std::cout << "位置无效！" << std::endl;**  **return false;**  **}**  **Node \*p = head->next;**  **for (int j = 1; j < i; j++) {**  **p = p->next;**  **}**  **e = p->data;**  **p->prev->next = p->next;**  **p->next->prev = p->prev;**  **delete p;**  **length--;**  **return true;**  **}**  **void DList::display() const {**  **Node \*p = head->next;**  **while (p != tail) {**  **std::cout << p->data << " ";**  **p = p->next;**  **}**  **std::cout << std::endl;**  **}**    **#endif**  **DStack.h**  **#ifndef DStACK\_H**  **#define DStACK\_H**  **#include "Dlist.h"**  **class DStack:public DList{**  **public:**  **DStack(); // 构造函数**  **using DList::DList;**  **~DStack() {}**  **// 入栈**  **bool push(int e);**  **// 出栈**  **bool pop(int &e);**  **// 获取栈顶元素**  **bool top(int &e) const;**  **// 判断栈是否为空**  **bool isEmpty() const;**  **// 获取栈的大小**  **int size() const;**  **};**  **DStack::DStack() : DList() {}**  **bool DStack::push(int e) {**  **return insert(1, e); // 在头部插入元素**  **}**  **bool DStack::pop(int &e) {**  **if (isEmpty()) {**  **std::cout << "栈为空，无法出栈！" << std::endl;**  **return false;**  **}**  **return remove(1, e); // 删除头部元素**  **}**  **bool DStack::top(int &e) const {**  **if (isEmpty()) {**  **std::cout << "栈为空，无法获取栈顶元素！" << std::endl;**  **return false;**  **}**  **return getElem(1, e); // 获取头部元素**  **}**  **bool DStack::isEmpty() const {**  **return DList::isEmpty(); // 调用基类的isEmpty方法**  **}**  **int DStack::size() const {**  **return DList::size(); // 调用基类的size方法**  **}**  **#endif**  **Main.cpp:**  **#include<iostream>**  **#include "include/DStack.h"**  **using namespace std;**  **int main() {**  **DStack stack;**  **int e;**  **std::cout << "测试栈基本操作：" << std::endl;**  **// 测试入栈操作**  **std::cout << "\n===== 测试入栈操作 =====" << std::endl;**  **stack.push(10);**  **stack.push(20);**  **stack.push(30);**  **stack.display();**  **// 测试获取栈顶元素**  **std::cout << "\n===== 测试获取栈顶元素 =====" << std::endl;**  **if (stack.top(e)) {**  **std::cout << "栈顶元素是：" << e << std::endl;**  **}**  **// 测试出栈操作**  **std::cout << "\n===== 测试出栈操作 =====" << std::endl;**  **if (stack.pop(e)) {**  **std::cout << "出栈元素是：" << e << std::endl;**  **}**  **stack.display();**  **// 测试其他操作**  **std::cout << "\n===== 测试其他操作 =====" << std::endl;**  **std::cout << "栈的大小：" << stack.size() << std::endl;**  **std::cout << "栈是否为空：" << (stack.isEmpty() ? "是" : "否") << std::endl;**  **// 测试清空操作**  **std::cout << "\n===== 测试清空操作 =====" << std::endl;**  **stack.clear();**  **std::cout << "清空后，栈的大小：" << stack.size() << std::endl;**  **std::cout << "清空后，栈是否为空：" << (stack.isEmpty() ? "是" : "否") << std::endl;**    **return 0;**  **}** | | | | | |
| 1. **实验结果与分析**   **本实验基于双向链表类 DList 成功派生出了栈类 DStack，并通过继承方式重用其基本链表操作函数，实现了以下栈操作功能：**   * **push(int e)：实现元素入栈（头插法）** * **pop(int &e)：实现栈顶元素出栈** * **top(int &e)：获取当前栈顶元素** * **isEmpty()：判断栈是否为空** * **size()：返回当前栈中元素数量**   **通过编写测试代码，对栈的常规使用场景进行测试，操作流程包括：**   1. **连续 push 若干元素；** 2. **使用 top 查看栈顶；** 3. **反复 pop 出栈并观察结果；** 4. **在空栈状态下尝试 pop 和 top，验证错误处理逻辑；** 5. **使用 isEmpty() 与 size() 检查栈状态。** | | | | | |
| 1. **实验心得**   通过本次实验，我深入理解了**类的继承机制**在实际编程中的应用价值。通过将 DList 作为基类，派生出 DStack 类，不仅减少了代码的重复编写，也增强了程序的可读性与扩展性。  在设计栈的各个成员函数时，我逐步体会到**面向对象编程思想的优势**，例如通过公有继承方式复用链表已有的插入、删除、获取元素等操作，从而高效地构建出符合“后进先出”特性的数据结构——栈。  此外，在测试过程中，我也更加重视**边界条件的处理**。例如在栈空时，不能盲目执行出栈或取栈顶操作，而应该先判断是否为空，这一过程锻炼了我对健壮性设计的意识。 | | | | | |
| 1. **教师评语** | | | | | |
| 1. **实验成绩**   教师签名： 蒋振刚 批阅日期： 2024 年 4 月 21日 | | | | | |

注：项目性质为 演示型、验证型、设计型、综合型和创新型。