《Java》课程实验报告

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 专业名称 | 计算机科学与技术 | 年级 | 大二 | 班级 | 计工本2302 |
| 学生姓名 | 朱会琛 | 学号 | 202311000202 | 指导教师 | 梁成 |
| 实验题目 | Java实验报告 | | | 提交时间 | 2024/11/21 |

1. 实验目的和要求

实验背景：

线性表的链式存储结构的特点是用一组任意的存储单元存储线性表的数据元素（这组存储单元可以连续，也可以不连续）。因此，为了表示每个数据元素*ai*与其直接后继数据元素*ai+*1之前的逻辑关系，对数据元素*ai*来说，除了存储其本身的信息之外，还需存储一个指示其后继的信息（即直接后继的存储位置）。这两部分信息组成数据元素*ai*的存储映像，称为结点（node）。它包括两个域：其中存储数据元素信息的域称为数据域；存储直接后继存储位置的域称为指针域。指针域中存储的信息称作指针或链。*n*个结点（*ai*(1 ≤ *i* ≤ *n*)的存储映像）链结成一个链表，即为线性表：

(*a*1, *a*2, ..., *an*)

单链表的可由头指针唯一确定，在C语言中可用“结构指针”来描述：

//---------------线性表的单链表存储结构------------------

typedef struct LNode {

ElemType data;

struct LNode \*next;

}LNode, \*LinkList

实验目的：

能够熟练运用Java中面向对象程序设计思路实现单链表

1. 实验环境

线上环境：idea + 记事本

线下环境：E315机房

三、实验内容及实施

**实验1：实验内容**

根据背景介绍中C语言版的单链表定义，给出相应的Java版本实现，并能够实现单链表的增、删、查找操作。具体要求如下：

1. 利用Java实现上述单链表的定义，并根据需要定义相应的构造函数。
2. 生成一个长度为*n*的单链表结构；定义函数打印该单链表。
3. 在指定位置添加元素，例如，用户输入两个整数0，2；表示在第0个位置插入data值为2的结点。其中第0个位置指的即为整个单链表的第1个元素位置（单链表中元素位置索引从0开始）。
4. 在指定位置删除元素，例如，用户输入一个整数1，表示删除单链表为位置索引为1的元素（即单链表的第二个元素）。

**备注：**

* 本次实验要撰写实验报告，并附程序运行结果截图
* 核心源代码需要贴入实验报告

**输出格式提示：**

**输出格式：**

System.out.println("请输入一个整数n，表示链表长度（n >= 2）：");

System.out.println("请依次输入链表数据：");

System.out.println("打印链表数据：");

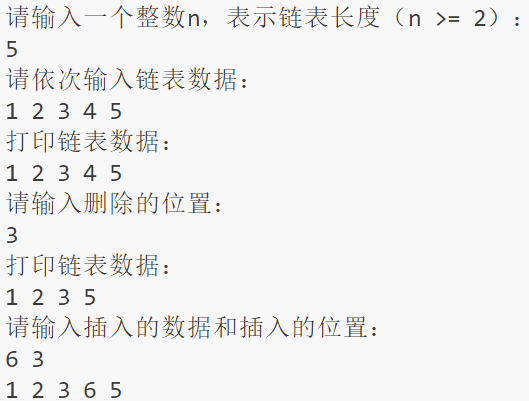
System.out.println("请输入删除的位置：");

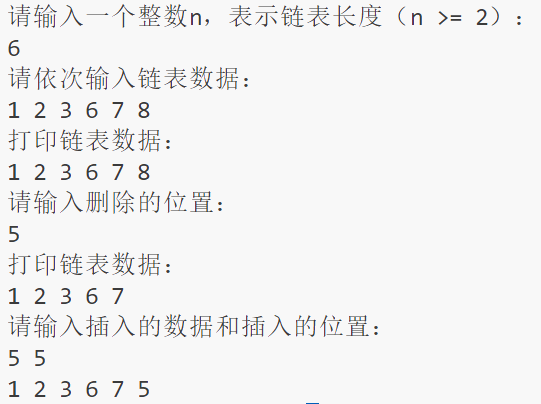
System.out.println("打印链表数据：");

System.out.println("请输入插入的数据和插入的位置：");

System.out.print(temp.data + " ");

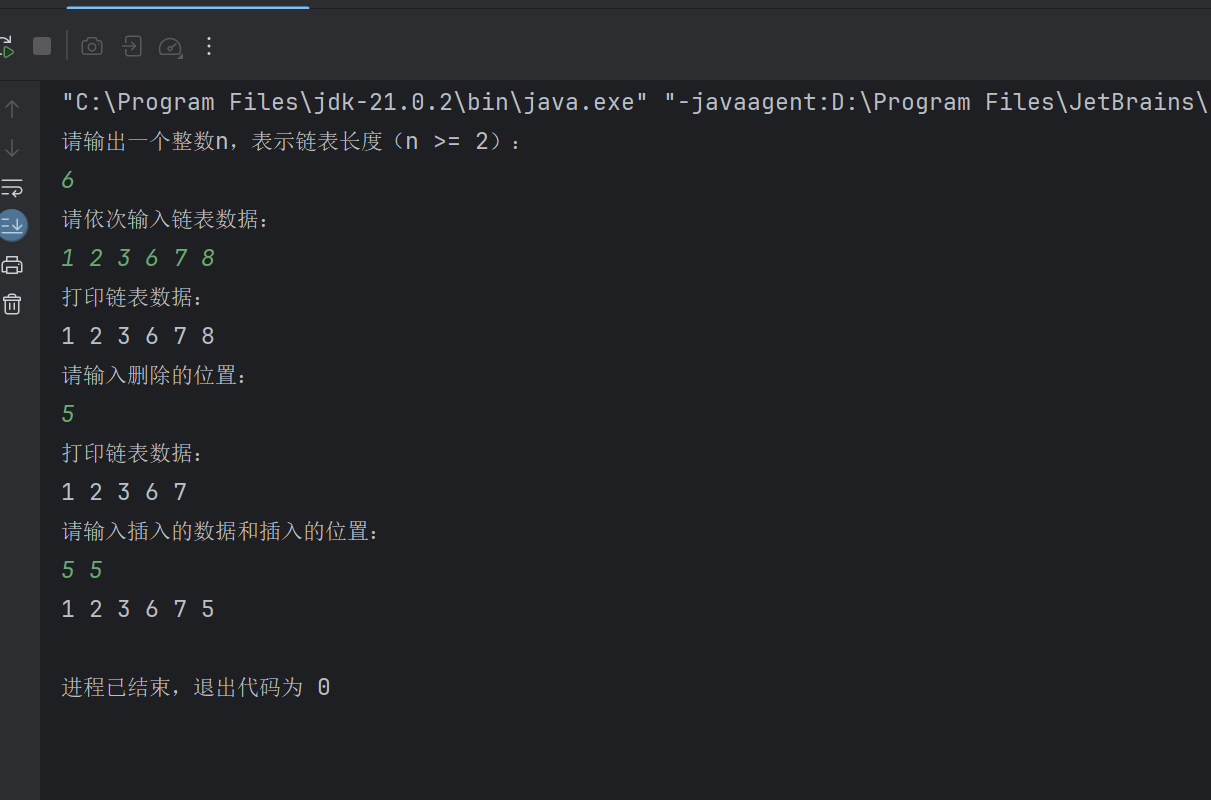
**下面是程序运行的2个样例（特别注意，必须要保证程序中的所有输出语句与测试用例中的完全一致才能正确通过平台测试，上传的代码不要有包名，否则可能导致结果错误）：**





1. **实验结果 (程序的执行结果)**

**运行结果：**

****

**四、实验源码：**

package Homework08**;**

**import** java**.**util**.**Scanner**;**

class Node **{**

int **data;**

Node next**;**

public Node**(**int **data) {**

this**.data** **=** **data;**

this**.**next **=** null**;**

**}**

**}**

class LinkedList **{**

private Node head**;**

Scanner sc **=** new Scanner**(**System**.in);**

**//** 初始化构造函数

public LinkedList**(){**

this**.**head **=** null**;**

**}**

**//** 创建

public void addLinkedList**(**int x**){**

**if** **(**x **<** **2){**

System**.**out**.**println**(**"链表长度必须大于等于**2**"**);**

return**;**

**}**

System**.**out**.**println**(**"请依次输入链表数据："**);**

**for** **(**int i **=** **0;**i **<** x**;**i**++){**

int val **=** sc**.**nextInt**();**

Node newNode **=** new Node**(**val**);**

**if** **(**head **==** null**){**

head **=** newNode**;**

**}** **else** **{**

Node temp **=** head**;**

**while** **(**temp**.**next **!=** null**){**

temp **=** temp**.**next**;**

**}**

temp**.**next **=** newNode**;**

**}**

**}**

**}**

**//** 删除元素

public void deleteLinkedList**(**int pos**){**

**if** **(**head **==** null**){**

System**.**out**.**println**(**"链表为空，无法删除！"**);**

return **;**

**}**

**if** **(**pos **==** **0){**

head **=** head**.**next**;**

**}** **else** **{**

Node temp **=** head**;**

**for** **(**int i **=** **0;**i **<** pos **-** **1;** i**++){**

**if** **(**temp **==** null || temp**.**next **==** null**){**

System**.**out**.**println**(**"删除位置无效**!**"**);**

return**;**

**}**

temp **=** temp**.**next**;**

**}**

**if** **(**temp**.**next **!=** null**){**

temp**.**next **=** temp**.**next**.**next**;**

**}**

**}**

**}**

**////** 查询元素

**//** public int queryLinkedList**(**int pos**){**

**//** Node newNode **=** head**;**

**//** **for** **(**int i **=** **0;**i **<** pos**;**i**++){**

**//** newNode **=** newNode**.**next**;**

**// }**

**//** return newNode**.data;**

**// }**

**//** 插入元素

public void insertLinkedList**(**int pos**,**int val**){**

Node newNode **=** new Node**(**val**);**

**if** **(**pos **==** **0){**

newNode**.**next **=** head**;**

head **=** newNode**;**

**}** **else** **{**

Node temp **=** head**;**

**for** **(**int i **=** **0;**i **<** pos **-** **1;**i **++){**

**if** **(**temp **==** null || temp**.**next **==** null**){**

System**.**out**.**println**(**"插入位置无效**!**"**);**

return **;**

**}**

temp **=** temp**.**next**;**

**}**

newNode**.**next **=** temp**.**next**;**

temp**.**next **=** newNode**;**

**}**

**}**

**//** 打印

public void display**(){**

Node temp **=** head**;**

**while** **(**temp **!=** null**){**

System**.**out**.print(**temp**.data** **+** " "**);**

temp **=** temp**.**next**;**

**}**

System**.**out**.**println**();**

**}**

**}**

public class LinkedListDemo **{**

public static void main**(**String**[]** args**) {**

Scanner sc **=** new Scanner**(**System**.in);**

LinkedList list **=** new LinkedList**();**

System**.**out**.**println**(**"请输出一个整数n，表示链表长度（n **>=** **2**）："**);**

int n **=** sc**.**nextInt**();**

list**.**addLinkedList**(**n**);**

System**.**out**.**println**(**"打印链表数据："**);**

list**.**display**();**

System**.**out**.**println**(**"请输入删除的位置："**);**

int deletePos **=** sc**.**nextInt**();**

list**.**deleteLinkedList**(**deletePos**);**

System**.**out**.**println**(**"打印链表数据："**);**

list**.**display**();**

System**.**out**.**println**(**"请输入插入的数据和插入的位置："**);**

int insertVal **=** sc**.**nextInt**();**

int insertPos **=** sc**.**nextInt**();**

list**.**insertLinkedList**(**insertPos**,**insertVal**);**

**//** System**.**out**.**println**(**"打印链表数据："**);**

list**.**display**();**

**}**

**}**

**五、实验讨论（可选）**

**答：**1.通过本次实验，我深入学习了单链表这一线性表的链式存储结构，并且在Java语言中实践了其实现与操作。单链表的存储特点在于通过指针域连接各个节点，从而实现数据的动态存储。相比于顺序存储结构，单链表在插入与删除操作上具有较高的效率，但在查找方面稍显不足。这种数据结构的灵活性使其在需要频繁修改数据的场景下更具优势。

2.实验过程中，我以面向对象的思想设计了Node和SingleLinkedList两个类。Node类作为节点的抽象，将数据域与指针域封装起来；SingleLinkedList类则负责实现链表的基本操作，包括创建、打印、插入和删除。通过合理设计构造函数与方法，代码的可读性与复用性得到了提升。这让我进一步体会到面向对象编程的优势。

3．在实验中，我特别关注了边界条件的处理。例如，对于链表为空或插入、删除位置非法的情况，程序能够输出提示并防止运行时错误。通过调试，我发现了链表操作中的一些细节问题，比如插入操作需特别小心链表头节点的特殊处理。

4.总之，这次实验让我更好地掌握了单链表的原理和实现，并通过代码实践增强了逻辑思维和编程能力。这为后续学习更复杂的数据结构奠定了扎实基础。