作业三 线性表的顺序表示

1、什么是线性表？

线性表（List）是数据结构的一种，它是零个或多个数据元素的有限序列。线性表的数据集合为 {a1,a2,…,an}，假设每个元素的类型均为DataType。其中，除第一个元素a1外，每一个元素有且只有一个直接前驱元素，除了最后一个元素an外，每一个元素有且只有一个直接后继元素。数据元素之间的关系是一对一的关系。

线性表可以有两种主要的存储结构：

* 顺序存储结构：用一组地址连续的存储单元依次存储线性表的数据元素，这种存储结构的线性表称为顺序表。
* 链式存储结构：用一组任意的存储单元存储线性表中的数据元素，称为线性表的链式存储结构。

线性表是最基本、最简单、也是最常用的一种数据结构。在实际应用中，常以栈、队列、字符串等特殊形式使用。

2、线性表的主要操作有哪些？

线性表的主要操作包括：

* InitList(&L)：初始化表。构造一个空的线性表。
* Length(L)：求表长。返回线性表L的长度，即L中数据元素的个数。
* LocateElem(L,e)：按值查找操作。在表L中查找具有给定关键字值的元素。
* GetElem(L,i)：按位查找操作。获取表L中第i个位置的元素的值。
* ListInsert(&L,i,e)：插入操作。在表L中的第i个位置上插入指定元素e。
* ListDelete(&L,i,&e)：删除操作。删除表L中的第i个位置的元素，并用e返回删除元素的值。
* PrintList(L)：输出操作。按前后顺序输出线性表L的所有元素值。
* Empty(L)：判空操作。若L为空表，则返回true，否则返回false。

以上就是线性表的基本操作，希望对你有所帮助！

3、什么是线性表的顺序存储？如何进行线性表的顺序存储？

线性表的顺序存储是一种数据存储方式，它将逻辑上相邻的元素存储在物理位置上也相邻的存储单元中。这种存储方式可以实现随机存取，即可以通过存储在线性表中的第一个数据元素的地址，找到任意位置的数据元素。

线性表的顺序存储通常使用一维数组来实现。例如，可以定义一个结构体，其中包含一个用于存放数据元素的数组和一个用于存放线性表长度的变量。

在 Java 中，可以使用数组来实现线性表的顺序存储。以下是一个简单的示例，它定义了一个名为 MyArrayList 的类，该类实现了线性表的基本操作：

public class MyArrayList {

private int[] array; // 存储数据的数组

private int size; // 线性表的当前大小

public MyArrayList(int capacity) {

array = new int[capacity];

size = 0;

}

// 插入元素

public void add(int index, int element) {

// 检查索引是否有效

if (index < 0 || index > size) {

throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + size);

}

// 如果数组已满，需要扩容

if (size >= array.length) {

int[] newArray = new int[array.length \* 2];

System.arraycopy(array, 0, newArray, 0, array.length);

array = newArray;

}

// 将元素从 index 开始向右移动一位

System.arraycopy(array, index, array, index + 1, size - index);

array[index] = element;

size++;

}

// 删除元素

public void remove(int index) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + size);

}

int numMoved = size - index - 1;

if (numMoved > 0) {

System.arraycopy(array, index + 1, array, index, numMoved);

}

size--;

}

// 获取元素

public int get(int index) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + size);

}

return array[index];

}

// 设置元素

public void set(int index, int element) {

if (index < 0 || index >= size) {

throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + size);

}

array[index] = element;

}

// 线性表的大小

public int size() {

return size;

}

// 判断线性表是否为空

public boolean isEmpty() {

return size == 0;

}

}

这个 MyArrayList 类实现了线性表的基本操作，包括添加元素、删除元素、获取元素、设置元素、获取线性表的大小以及判断线性表是否为空等功能。

4、请解释以下代码的含义：

public class Student {

private int id;

private String name;

private char gender;

private int age; //年龄

private String nativeplace;

public Student(){ }

public Student(int id, String name){

this.id=id;

this.name= name;}

public int getId() {

return id;}

public void setId(int id) {

this.id = id;}

public String getName() {

return name;}

public void setName(String name){

this.name = name;}

public char getGender() {

return gender;}

public void setGender(char gender) {

this.gender = gender;}

public int getAge() {

return age;}

public void setAge(int age) {

this.age= age;}

public String getNativeplace() {

return nativeplace;}

public void setNativeplace(String nativeplace){

this.nativeplace= nativeplace;}

}

public class StuSeqList {

private Student[] students;

private int maxlength;

private int length;

public StuSeqList(int maxlength) {

initiate(maxlength);

}

private void initiate(int maxlength){

students = new Student[maxlength];

this.maxlength = maxlength;

}

public int length() {

return length;}

public int maxlength() {

return maxlength;}

}

这段代码定义了两个类：Student 和 StuSeqList。

Student 类代表一个学生，具有以下属性：

* id：学生的 ID。
* name：学生的名字。
* gender：学生的性别。
* age：学生的年龄。
* nativeplace：学生的籍贯。
* Student 类还包含了一些方法，用于获取或设置这些属性的值（Getter/Setter）。

StuSeqList 类代表一个学生列表，具有以下属性：

* students：一个 Student 对象的数组，用于存储学生信息。
* maxlength：列表的最大长度。
* length：当前列表的长度。

StuSeqList 类的构造函数接受一个 maxlength 参数，用于初始化 students 数组和 maxlength 属性。此外，StuSeqList 类还包含了一些方法，用于获取列表的当前长度和最大长度。这个类可以用于管理一个学生列表，例如添加新的学生，删除学生，查找学生等。但是，这段代码并没有提供这些功能，需要自己实现。

5、请简述顺序线性表是如何实现数据插入操作的。

顺序线性表的数据插入操作通常遵循以下步骤：

* 检查空间：首先，检查线性表是否有足够的空间来插入新的元素。如果没有，可能需要扩大数组的大小。
* 移动元素：找到要插入新元素的位置，将该位置及其后面的所有元素向后移动一位。这样可以在指定位置腾出空间来插入新的元素。
* 插入元素：在腾出的空位上插入新的元素。
* 更新长度：最后，更新线性表的长度。

这是一个基本的插入操作，实际的实现可能会根据具体的需求和环境有所不同。例如，如果线性表是一个动态数组，当空间不足时，可能需要重新分配更大的内存空间。

6、请解释以下代码的含义：

public boolean insert(int index, Student student) {

boolean suc = false;

if (index <0 || index> length) {

System.out.println("插入位置出错!");

return suc;}

if (length + 1> maxlength) {

System.out.println("线性表已满!");

return suc;}

length++;

for (; index < length; index++) {

Student stunext = students[index];

students[index] = student;

student = stunext;

suc = true;

}

return suc;

}

这段代码是一个插入操作的函数，用于在特定位置插入一个学生对象到一个学生数组中。以下是详细的步骤：

函数首先定义一个布尔变量suc，初始值为false，用于表示插入操作是否成功。

然后，函数检查插入的位置index是否在有效范围内（即0到数组长度length之间）。如果不在有效范围内，函数会打印出"插入位置出错!"的错误信息，并返回suc（此时为false）。

如果插入的位置有效，函数接着检查数组是否已满（即数组长度length加1是否大于数组的最大长度maxlength）。如果数组已满，函数会打印出"线性表已满!"的错误信息，并返回suc（此时为false）。

如果插入的位置有效且数组未满，函数会增加数组的长度length。

然后，函数会从插入的位置开始，将每个位置的学生对象向后移动一位，为新的学生对象腾出空间。

最后，函数将新的学生对象插入到指定的位置，并将suc设为true，表示插入操作成功。

函数返回suc，表示插入操作是否成功。

这个函数的主要用途是在特定位置插入一个新的学生对象，如果插入成功，返回true，否则返回false。这是数据结构中线性表插入元素的常见操作。

7、请简述顺序线性表是如何实现数据删除操作的。

顺序线性表的数据删除操作通常包括以下步骤：

定位：首先，我们需要找到要删除的元素在线性表中的位置。这通常通过比较元素的键值来实现。

检查：在找到元素的位置后，我们需要检查该位置是否存在。如果不存在，那么删除操作就无法进行。

删除：如果元素存在，我们就可以进行删除操作。删除操作通常是通过将位置后面的所有元素向前移动一位来实现的。这样，原来的元素就被覆盖，从而实现了删除。

更新长度：最后，我们需要更新线性表的长度。因为我们删除了一个元素，所以线性表的长度应该减少1。

8、请解释以下代码的含义：

public boolean delete(int index){

boolean suc = false;

if (index < 0 || index >= length) {

System.out.println("删除位置出错!");

return suc;}

for (; index< length-1; index++) {

students[index] = students[index+1];

suc = true;}

length--;

return suc;

}

这段代码是一个删除操作的函数，用于从学生数组中删除特定位置的学生对象。以下是详细的步骤：

函数首先定义一个布尔变量suc，初始值为false，用于表示删除操作是否成功。

然后，函数检查要删除的位置index是否在有效范围内（即0到数组长度length之间）。如果不在有效范围内，函数会打印出"删除位置出错!"的错误信息，并返回suc（此时为false）。

如果要删除的位置有效，函数会从该位置开始，将后面的所有学生对象向前移动一位，覆盖掉要删除的学生对象。

然后，函数将suc设为true，表示删除操作成功。

最后，函数减少数组的长度length，并返回suc，表示删除操作是否成功。

这个函数的主要用途是删除特定位置的学生对象，如果删除成功，返回true，否则返回false。这是数据结构中线性表删除元素的常见操作。