

<http://www.cnblogs.com/smyhvae/p/4758808.html>

本节内容：

- 线性结构
- 线性表抽象数据类型
- 顺序表
- 顺序表应用

### 一、线性结构：

如果一个数据元素序列满足：

- (1) 除第一个和最后一个数据元素外，每个数据元素只有一个前驱数据元素和一个后继数据元素；
- (2) 第一个数据元素没有前驱数据元素；
- (3) 最后一个数据元素没有后继数据元素。

则称这样的数据结构为线性结构。

### 二、线性表抽象数据类型(ADT)：

#### 1、线性表抽象数据类型的概念：

线性表抽象数据类型主要包括两个方面：既数据集合(data)和该数据集合上的操作集合(operation)。

数据集合：

可以表示为 $a_0, a_1, a_2, \dots, a_{n-1}$ , 每个数据元素的数据类型可以是任意的类型。

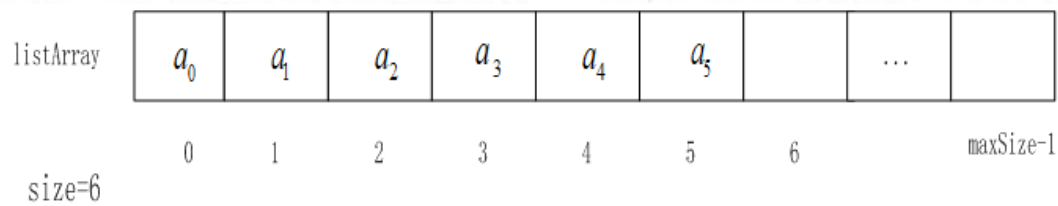
操作集合包括如下：

1. 求元素个数
2. 插入
3. 删除
4. 查找
5. 判断是否为空

### 三、顺序表：（在物理存储结构上连续，大小固定）

#### 1、顺序表的概念：

计算机有两种基本的存储结构（物理存储结构）：顺序结构、离散结构。使用顺序结构实现的线性表称为顺序表。如下图所示：



Java内存中，栈内存和堆内存占了很大一部分空间：栈内存的存储是顺序结构，堆内存的存储是离散结构。

## 2、设计顺序表类：

### (1) List.java

//线性表接口

```
public interface List {
    //获得线性表长度
    public int size();
    //判断线性表是否为空
    public boolean isEmpty();
    //插入元素
    public void insert(int index, Object obj) throws Exception;
    //删除元素
    public void delete(int index) throws Exception;
    //获取指定位置的元素
    public Object get(int index) throws Exception;
}
```

### (2) SequenceList.java: (核心代码)

```
1 public class SequenceList implements List {
2
3     //默认的顺序表的最大长度
4     final int defaultSize = 10;
5     //最大长度
6     int maxSize;
7     //当前长度
8     int size;
9     //对象数组
10    Object[] listArray;
11
12
13    public SequenceList() {
14        init(defaultSize);
15    }
16
17    public SequenceList(int size) {
18        init(size);
```

```
19     }
20
21     //顺序表的初始化方法
22     private void init(int size) {
23         maxSize = size;
24         this.size = 0;
25         listArray = new Object[size];
26     }
27
28     @Override
29     public void delete(int index) throws Exception {
30         // TODO Auto-generated method stub
31         if (isEmpty()) {
32             throw new Exception("顺序表为空，无法删除！");
33         }
34         if (index < 0 || index > size - 1) {
35             throw new Exception("参数错误！");
36         }
37         //移动元素
38         for (int j = index; j < size - 1; j++) {
39             listArray[j] = listArray[j + 1];
40         }
41         size--;
42     }
43
44     @Override
45     public Object get(int index) throws Exception {
46         // TODO Auto-generated method stub
47         if (index < 0 || index >= size) {
48             throw new Exception("参数错误！");
49         }
50         return listArray[index];
51     }
52
53     @Override
54     public void insert(int index, Object obj) throws Exception {
55         // TODO Auto-generated method stub
56         //如果当前线性表已满，那就不允许插入数据
57         if (size == maxSize) {
58             throw new Exception("顺序表已满，无法插入！");
59         }
60         //插入位置编号是否合法
61         if (index < 0 || index > size) {
62             throw new Exception("参数错误！");
63         }
64     }
65 }
```

```

64         //移动元素
65         for (int j = size - 1; j >= index; j--) {
66             listArray[j + 1] = listArray[j];
67         }
68         //不管当前线性表的size是否为零，这句话都能正常执行，即都能正常插入
69         listArray[index] = obj;
70         size++;
71
72     }
73
74     @Override
75     public boolean isEmpty() {
76
77         return size == 0;
78     }
79
80     @Override
81     public int size() {
82
83         return size;
84     }
85 }

```

我们来看一下第54行的插入操作insert()方法：如果需要在index位置插入一个数据，那么index后面的元素就要整体往后移动一位。这里面需要特别注意的是：

**插入操作：移动元素时，要从后往前操作，不能从前往后操作，不然元素会被覆盖的。**

**删除元素：移动元素时，要从前往后操作。**

### (3) 测试类：

```

public class Test {
    public static void main(String[] args) {
        SequenceList list = new SequenceList(20);
        try {
            list.insert(0, 100);
            list.insert(0, 50);
            list.insert(1, 20);

            for (int i = 0; i < list.size; i++) {
                System.out.println("第" + i + "个数为" + list.get(i));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

}

我们要注意插入的规则是什么，不然会觉得这个顺序表打印输出的顺序很奇怪。

运行效果：

```
Run Test
C:\Java\jdk1.7.0_71\bin\java ...
第0个数为50
第1个数为20
第2个数为100
Process finished with exit code 0
```

### 3、顺序表效率分析：

- 顺序表插入和删除一个元素的时间复杂度为 $O(n)$ 。
- 顺序表支持随机访问，顺序表读取一个元素的时间复杂度为 $O(1)$ 。因为我们是可以通过下标直接访问的，所以时间复杂度是固定的，和问题规模无关。

### 4、顺序表的优缺点：

- 顺序表的优点是：支持随机访问；空间利用率高（连续分配，不存在空间浪费）。
- 顺序表的缺点是：**大小固定**（一开始就要固定顺序表的最大长度）；插入和删除元素需要移动大量的数据。

### 5、顺序表的应用：

设计一个顺序表，可以保存100个学生的资料，保存以下三个学生的资料，并打印输出。

学号	姓名	性别	年龄
S00001	张三	男	18
S00002	李四	男	19
S00003	王五	女	21

#### (1)List.java:

和上面的代码保持不变

#### (2)SequenceList.java:

和上面的代码保持不变

#### (3)Students.java:学生类

//学生类

```
public class Students {
```



```

private String id;// 学号
private String name;// 姓名
private String gender;// 性别
private int age;// 年龄

public Students() { }

public Students(String sid, String name, String gender, int age) {
    this.id = sid;
    this.name = name;
    this.gender = gender;
    this.age = age;
}

public String getId() {
    return id;
}

public void setId(String id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}

public int getAge() {
    return age;
}

public void setAge(int age) {
    this.age = age;
}

public String toString() {
    return "学号: " + this.getId() + " 姓名: " + this.getName()
        + " 性别: " + this.getGender() + " 年龄:" + this.getAge();
}
}

```

(4)Test.java:

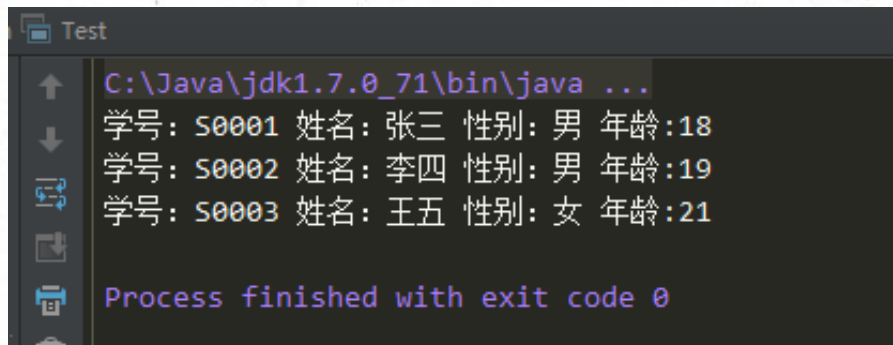
```

1 public class Test {
2
3     /**
4      * @param args
5      */
6     public static void main(String[] args) {
7
8         SequenceList list = new SequenceList(100);
9
10        try {
11            //第一个参数list.size代表的是：我每次都是在顺序表的最后一个
12            //位置(当前线性表的长度的位置)进行插入操作。这一行里，size是等于0
13            list.insert(list.size, new Students("S0001", "张三", "男",
14            18));
15            list.insert(list.size, new Students("S0002", "李四", "男",
16            19));
17            list.insert(list.size, new Students("S0003", "王五", "女",
18            21));
19
20            for (int i = 0; i < list.size; i++) {
21                System.out.println(list.get(i));
22            }
23        } catch (Exception ex) {
24            ex.printStackTrace();
25        }
26    }
27 }

```

注意第11行的注释：第一个参数list.size代表的是：我每次都是在顺序表的最后一个位置（当前线性表的长度的位置）进行插入操作；这样的话，遍历时才是按照张三、李四、王五的顺序进行输出的。

运行效果：



```

Test
C:\Java\jdk1.7.0_71\bin\java ...
学号: S0001 姓名: 张三 性别: 男 年龄:18
学号: S0002 姓名: 李四 性别: 男 年龄:19
学号: S0003 姓名: 王五 性别: 女 年龄:21
Process finished with exit code 0

```