线性表

> 线性表是线性结构的一种，其基本操作主要有获取元素值，设置，遍历，插入，查找，替换，排序等等。

定义（Linear List)

> 有n（n >= 0）个类型相同的数据组成的有限序列，其中元素的数据类型可以是整数，浮点数，字符串，亦或者是类。

表达式

$$a1，a2，a3，，，a（i-1），a（i），a（i+1），，，a（n-1）$$

元素名称

> 1. 前驱元素：a（i - 1）
> 2. 后置元素：a（a+1）
> 3. 记录：

```
[{
    name:"haba",     //一个数据元素由多个数据项组成。
    age:20
}]
```

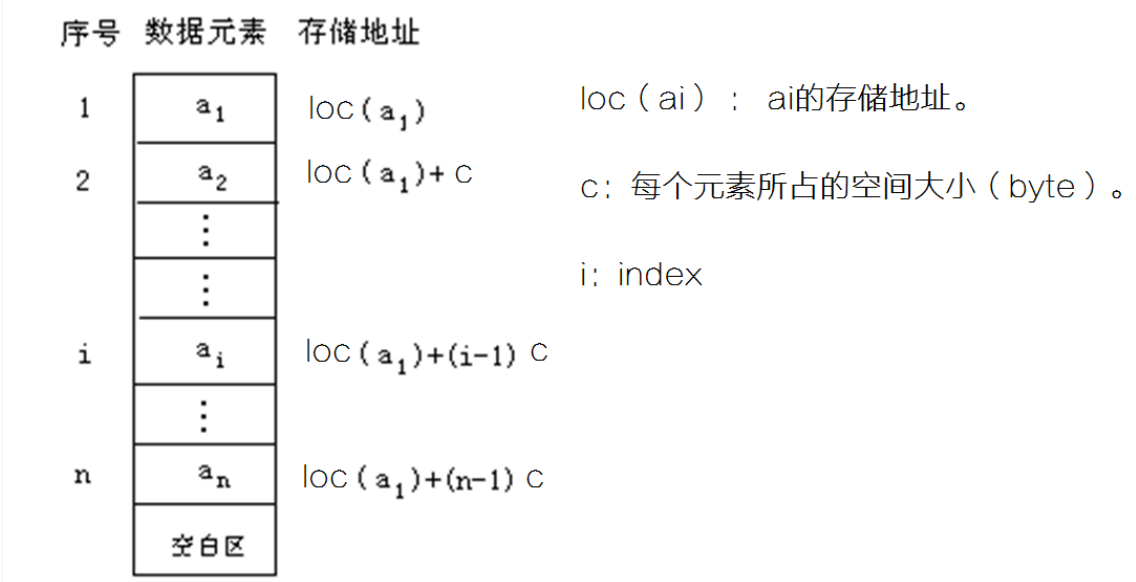
特性

> 存在唯一的最后一个元素。
>
> 存在唯一的首个元素。
>
> 第一个元素无前驱元素，最后一个元素无后继元素，
>
> 除了第一个和最后一个，每一个元素都只有一个前驱元素和一个后继元素。

储存结构

顺序存储结构：

> 属于一种**随机访问结构**
>
> **随机访问结构**指可以随机访问任意元素，在你的代码中，你只需要提供一个index，程序自动帮你处理返回。



顺序存储抽象数据类型（ADT）

代码中我们用到的比较常用的线性存储结构就是数组了，我们还可以试着自己抽象数据类型实现线性表。

```
public class SeqList<T> extends Object{      //T表示线性表数据元素类型
    protected Object[] element;      //对象数组储存数据元素，保证成员类型不受限制。
    protected int n;                 //这里声明顺序表元素个数。
    public SeqList(int length)       //创建容量为length的空表
    {
        this.element = new Object[length];  //声明万金油数组最大容量
        this.n = 0;                         //实际长度。
    }
    public SeqList()
    {
        this(64);       //创建默认容量的空表，构造方法重载。
    }
    public SeqList(T[] values)       //填充元素
    {
        this(values.length);
        for(int i = 0;i < values.length;i++){
            this.element[i] = values[i]
        }
        this.n = element.length;
    }
    public boolean isEmpty()
    {
        return this.n == 0;
    }
    public int size()
    {
        return this.n;
    }
    public T get(int i)
    {
        if(i >= 0 && i < this.n){
            return (T)this.element[i];  // ????
        }
        return null;
    }
```

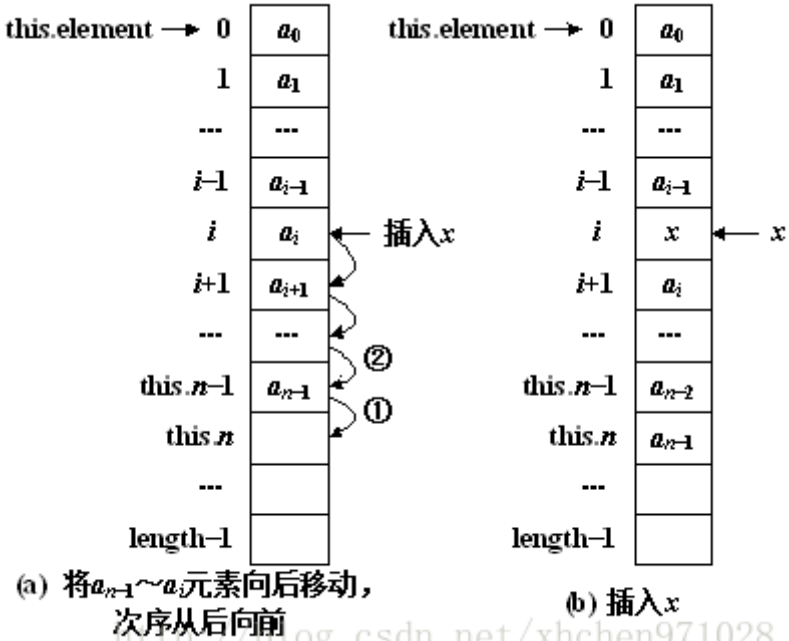
```
    public void set(int i,T x)
    {
        if(x==null){
            throw new NullPointerException("x == null");
        }
        if(i>=0 && i<this.n){
            this.element[i] = x;        //存入element万金油数组，更新数据元素。
        }else{
            throw new java.lang.IndexOutOfBoundsException(i+"")
        }
    }
    public String toString(){
        String str = this.getClass().getName()+"(";
        if(this.n >= 0){
            str += this.element[0].toString();
            for(int i = 1;i<this.n;i++){
                str += this.element[i].toString();  //调用T类的toString，运行时多态。
            }
        }
        return str+")";
    }
}
```

> 以上为比较简单的抽象类型声明方法，以下为逻辑较复杂的插入，删除等。

插入

插入操作我们需要考虑的主要有以下几点：

> 1.元素移动，要将元素依次向后移动一位。
>
> 2.之后我们要考虑储存线性表数据元素的储存空间大小，如果超出了我们预先申请的**Object**数组长度，就会造成**数据溢出**，而我们需要的就是在**数据溢出**之前，申请一个更大容量的数组，并且赋值数组元素。




```
public int insert(int i,T x)
{
```

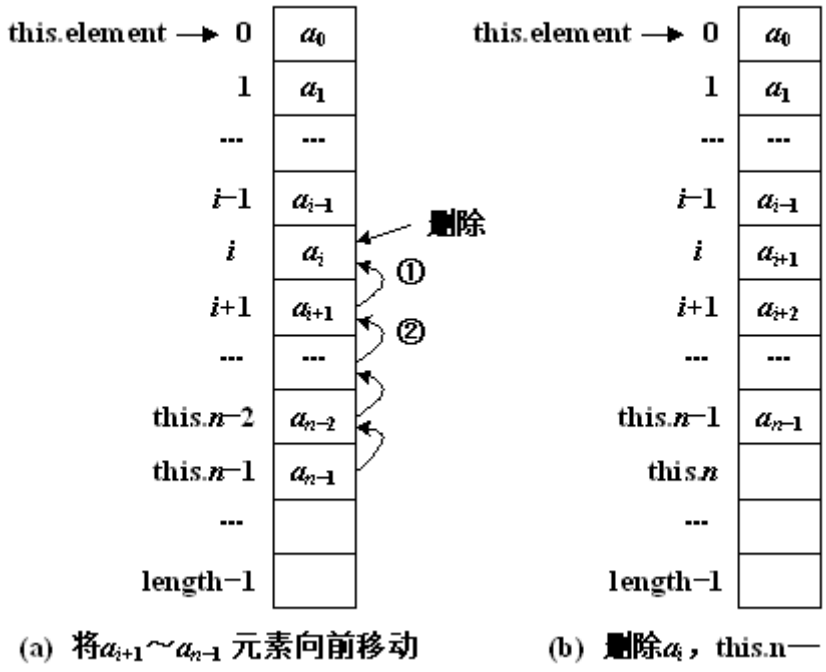
```
    if(i > this.n) i = n;
    if(i < 0)i = 0;
    Object[] source = this.element;      //赋值数据元素数组。
    if(this.element.length == this.n){
        this.element = new Object[source.length*2];
        for(int j = 0; j < i; j++){
            this.element[j] = source[j];
        }
        this.element[i] = x;
        for(int j = this.n-1; j >= i ;j--){
            this.element[j+1] = source[j];
        }
        this.n++;
        return i;
    }
}

public int insert(T x){
    return this.insert(this.n,x)
}
```

粗略性能分析

> 插入内容：0 1 2 3 4 ..... n-1 n
>
> 移动次数：n n-1 n-1 n-3 .... 1 0
>
> 平均移动次数：n/2

删除

删除嘛，肯定是比插入要简单的，只需要实现一个元素前移一位的操作就行了。



```
public T remove(int i){
    if(i >= 0&&i <= this.n){
        T oldELe = this.element[i];
        for(int j = i+1;j < this.n;j++){
            this.element[j] = this.element[j]
        }
        this.element[this.n - 1] = null;
        this.n--
        return oldEle
    }
    return null
}
```