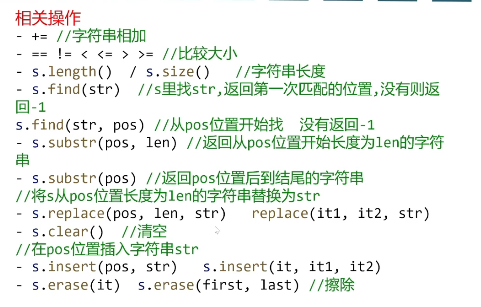
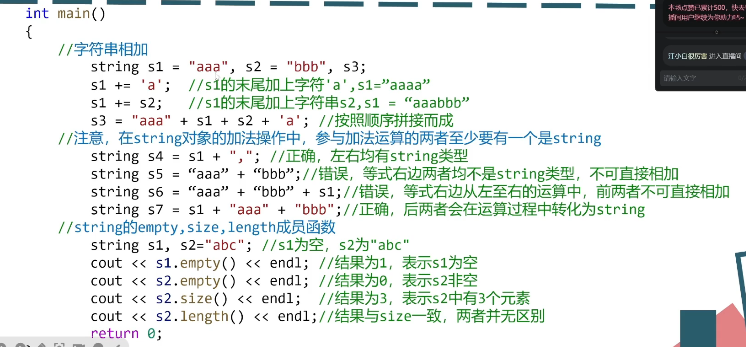
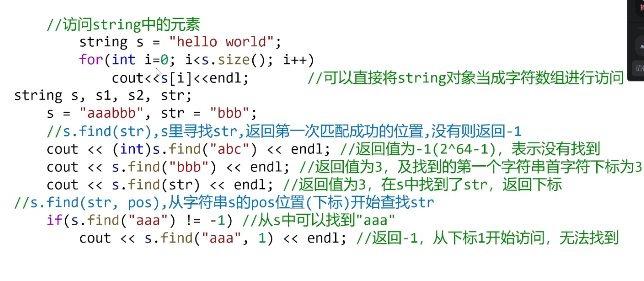
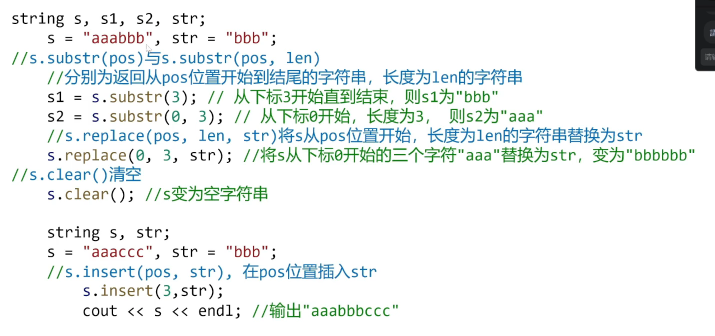
蛮力法：O(N^2)

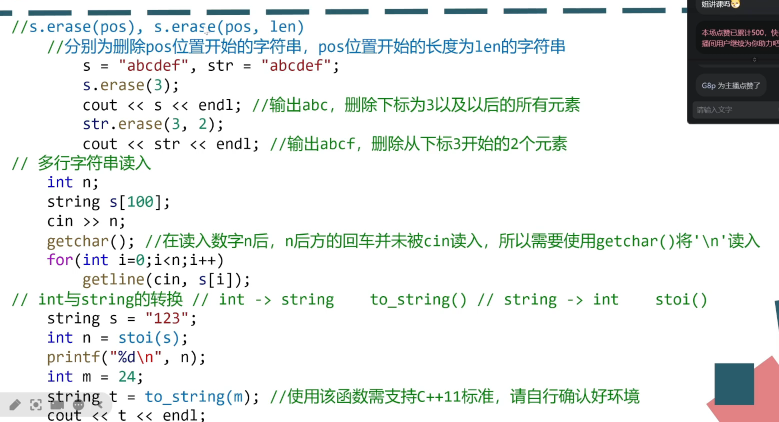
# STL库

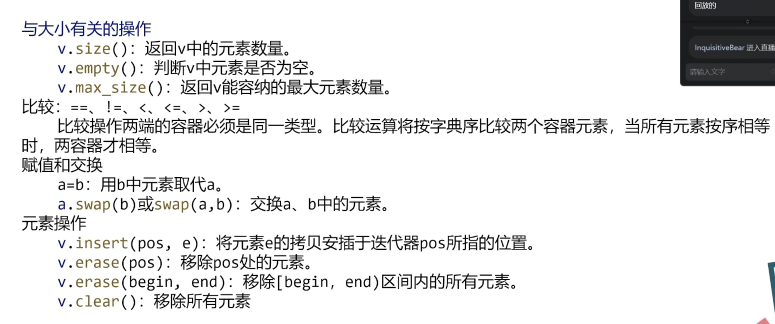












## Java的String库

StringBuilder

不会创建新的对象，而是直接在原对象上修改，因此在拼接或修改字符串时，它比 String 更快。

**常用方法**：

* append(String s): 在末尾追加字符串
* insert(int offset, String s): 在指定位置插入字符串
* delete(int start, int end): 删除指定范围内的字符
* replace(int start, int end, String s): 替换指定范围内的字符
* reverse(): 反转字符串
* toString(): 转换为 String
* charAt(int index)：它用于获取字符串中指定索引位置的字符

import java.lang.Character

可用于判断类型是数字还是字符（char）

eg:Character.isLetter(x);判断x是不是字母

Character.isDigit(x);判断x是不是数字

## Array

Lambda表达式

(a, b) -> (b + a).compareTo(a + b)

 的作用是比较两个字符串 a 和 b，并决定它们的顺序。

(b + a) 表示将 b 和 a 连接起来，形成一个新的字符串（例如 "31213"）。

(a + b) 表示将 a 和 b 连接起来，形成另一个新的字符串（例如 "13312"）。

(b + a).compareTo(a + b) 是比较这两个新字符串的大小

Arrays.sort 是稳定的排序算法，这意味着如果两个元素的比较结果为相等，它们的相对顺序不会改变。因此，我们可以利用这一点来实现题目要求。

Integer.parseInt() 是 Java 中的一个常用方法，用于将字符串（String）转换为整数（int）。它是 Integer 类的一个静态方法，通常用于处理用户输入或从文件、网络等来源读取的字符串数据，将其转换为整数以便进行数学运算或其他操作。

### **语法**

java复制

int result = Integer.parseInt(String s);

****参数****：

s：要转换的字符串。字符串必须是一个有效的整数表示形式，例如 "123"、"-456" 等。

****返回值****：

返回一个 int 类型的整数，表示转换后的结果。

****异常****：

如果字符串不是有效的整数格式，会抛出 NumberFormatException 异常。

### **示例**

以下是一些使用 Integer.parseInt() 的示例：

#### **示例 1：正常转换**

java复制

String str = "123";int num = Integer.parseInt(str);System.out.println("转换结果: " + num); *// 输出：转换结果: 123*

#### **示例 2：负数转换**

String str = "-456";int num = Integer.parseInt(str);System.out.println("转换结果: " + num); *// 输出：转换结果: -456*

#### **示例 3：包含非数字字符**

String str = "abc";try {

int num = Integer.parseInt(str);

System.out.println("转换结果: " + num);} catch (NumberFormatException e) {

System.out.println("错误: " + str + " 不是有效的整数格式");}*// 输出：错误: abc 不是有效的整数格式*

#### **示例 4：空字符串**

String str = "";try {

int num = Integer.parseInt(str);

System.out.println("转换结果: " + num);} catch (NumberFormatException e) {

System.out.println("错误: 空字符串不是有效的整数格式");}*// 输出：错误: 空字符串不是有效的整数格式*

### **注意事项**

****合法性检查****：

在调用 .parseIntInteger() 之前，最好对字符串进行合法性检查，确保它是一个有效的整数格式。可以通过正则表达式或其他方式提前验证。

如果不确定字符串是否为合法的整数格式，建议使用 try-catch 块来捕获 NumberFormatException 异常，避免程序崩溃。

****范围限制****：

Integer.parseInt() 转换后的结果是一个 int 类型的整数，其范围是 -2,147,483,648 到 2,147,483,647。如果字符串表示的数字超出了这个范围，会抛出 NumberFormatException。

****空字符串处理****：

空字符串（""）不是有效的整数格式，调用 Integer.parseInt("") 会抛出 NumberFormatException。

### **替代方法**

如果需要更灵活的数字转换，可以使用其他方法：

**Integer.valueOf(String s)**：与 Integer.parseInt() 类似，返回但的是 Integer 对象，而不是原始的 int 类型。

**Double.parseDouble(String s)**：用于将字符串转换为 double 类型的浮点数。

**Long.parseLong(String s)**：用于将字符串转换为 long 类型的整数，适合处理更大的整数值。

总之，Integer.parseInt() 是一个非常实用的方法，但在使用时需要注意字符串的格式和合法性，以避免运行时错误。

关于题解：

if (sc.hasNextInt()) {

String[] inputNumbers = sc.nextLine().split(" ");

if (inputNumbers.length < n) {

return;

}

for (int i = 0; i < n; i++) {

numbers[i] = inputNumbers[i];

}

}

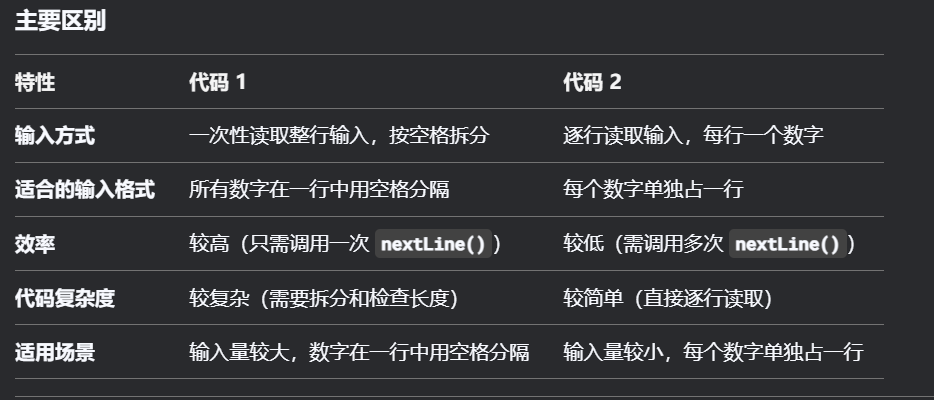
sc.close();

和 if (sc.hasNextInt()) {

for(int i = 0 ; i < n;i++) {

numbers[i] = sc.nextLine();

}有什么区别？

也可以：String line = sc.nextLine().trim(); // 读取整行并去除前后空格

**a[1].compareTo(b[1])适用于字符串**

**Integer**.compare(a[1],b[1])适用于整数

### Java输入中split的用法

## Java中栈和队列的操作

## Java中Map函数如何使用

**HASH MAP:**指 key 到value的一个映射，value可以是null

**TREE MAP： ~,**treemap涉及到排序,不允许键为null

定义hash map举例：

`**HashMap<类型，类型> map = new HashMap<>()**`;

### 常用函数：

1. **Put(类型，类型) 放置数据**
2. **Get（类型key） 传入key可以得到对应的value**
3. **Containskey(类型 key) 查看是否包含该key，返回值类型boolean。**
4. **Containsvalue(类型 value) 同上**
5. **putAll :传入一个map集合,键值类型与传入map的相同**
6. **Replace(旧值（key），新值(value))**
7. **Replace(key,旧值（value），新值（value）)**
8. **Values 返回所有值**

**用法举例：**

**Collection<String> list = map1.values();//将所有的值包装成Collection返回**

**Iterator<String> it = list.iterator();**

**While(it.hasNext()){**

**System.out.println(it.next());//遍历输出**

**}**

1. **KeySet()**

**用法举例：**

**（ ps: set指返回一个不重复的值 ）**

**Set<Integer> Set = map1.KeySet();**

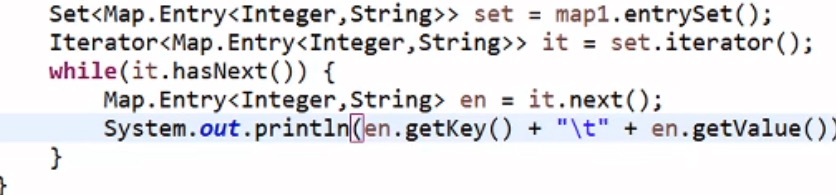
**Iterator<Integer> it = Set.iterator();**

**While(it.hasNext()){**

**System.out.println(it.next());//遍历输出**

**}**

1. **.EntrySet(可同时遍历kry,value两个数据)**

****

**关于TreeMap(用法与HashMap 80%相同)**

**用法举例：**

**TreeMap<Integer,String> map2 = new TreeMap<Integer,String>()**

## MATH函数

Integer.MAX\_VALUE 来自 **Java 标准库中的** Integer **类**，它表示 int **类型的最大值，**定义为 2147483647（即 2^31 - 1）。

# java中的高精度算法

**Import.java.math.BigInteger**

1. 构造方法

****从字符串创建****：通过 BigInteger(String val)，可以将一个字符串表示的数字转换为 BigInteger 对象。

BigInteger bigInt1 = new BigInteger("123456789012345678901234567890");

****从字节数组创建****：通过 BigInteger(byte[] val)，可以将一个字节数组转换为 BigInteger 对象。字节数组的最高字节是符号位，正数为正，负数为负。

byte[] bytes = {0x00, 0x01, 0x02, 0x03};BigInteger bigInt2 = new BigInteger(bytes);

1. 常用方法：

#### ****算术运算****

****加法****：add(BigInteger val)，返回两个 BigInteger 的和。

java复制

BigInteger result = bigInt1.add(bigInt2);

****减****法：subtract(BigInteger val)，返回两个 BigInteger 的差。

java复制

BigInteger result = bigInt1.subtract(bigInt2);

****乘法****：multiply(BigInteger val)，返回两个 BigInteger 的乘积。

BigInteger result = bigInt1.multiply(bigInt2);

****除法****：divide(BigInteger val)，返回两个 BigInteger 的商。需要注意的是，如果除数为零，会抛出 ArithmeticException。

BigInteger result = bigInt1.divide(bigInt2);

****取模****：mod(BigInteger m)，返回两个 BigInteger 的模。

java复制

BigInteger result = bigInt1.mod(bigInt2);

****幂运算****：pow(int exponent)，返回 BigInteger 的指数次幂。

java复制

BigInteger result = bigInt1.pow(3);

#### ****比较方法****

****比较大小****：compareTo(BigInteger val)，比较两个 BigInteger 的大小。如果调用对象大于参数对象，返回 1；相等返回 0；小于返回 -1。

java复制

int comparison = bigInt1.compareTo(bigInt2);

****判断是否相等****：equals(Object x)，判断两个 BigInteger 是否相等。

java复制

boolean isEqual = bigInt1.equals(bigInt2);

****获取绝对值****：abs()，返回 BigInteger 的绝对值。

java复制

BigInteger absValue = bigInt1.abs();

****取反****：negate()，返回 BigInteger 的相反数。

BigInteger negValue = bigInt1.negate();

****最大值和最小值****：max(BigInteger val) 和 min(BigInteger val)，分别返回两个 BigInteger 中的较大值和较小值。

BigInteger max = bigInt1.max(bigInt2);BigInteger min = bigInt1.min(bigInt2);

### ****进制转换****

BigInteger 支持不同进制的转换：

****从字符串创建指定进制的数字****：BigInteger(String val, int radix)，可以指定字符串的进制。

BigInteger hexValue = new BigInteger("1A3F", 16); *// 16进制*BigInteger binValue = new BigInteger("101010", 2); *// 2进制*

****将** BigInteger **转换为指定进制的字符串****：toString(int radix)，将 BigInteger 转换为指定进制的字符串。

String hexString = bigInt1.toString(16);

String binString = bigInt1.toString(2);

以下是一个完整的示例，展示 BigInteger 的常见用法：

import java.math.BigInteger;

public class BigIntegerExample {

public static void main(String[] args) {

*// 创建 BigInteger 对象*

BigInteger bigInt1 = new BigInteger("123456789012345678901234567890");

BigInteger bigInt2 = new BigInteger("987654321098765432109876543210");

*// 算术运算*

BigInteger sum = bigInt1.add(bigInt2);

BigInteger difference = bigInt1.subtract(bigInt2);

BigInteger product = bigInt1.multiply(bigInt2);

BigInteger quotient = bigInt1.divide(bigInt2);

BigInteger modulus = bigInt1.mod(bigInt2);

*// 输出结果*

System.out.println("Sum: " + sum);

System.out.println("Difference: " + difference);

System.out.println("Product: " + product);

System.out.println("Quotient: " + quotient);

System.out.println("Modulus: " + modulus);

*// 比较*

int comparison = bigInt1.compareTo(bigInt2);

System.out.println("Comparison: " + comparison);

*// 进制转换*

String hexString = bigInt1.toString(16);

System.out.println("Hexadecimal: " + hexString);

*// 绝对值和取反*

BigInteger absValue = bigInt1.abs();

BigInteger negValue = bigInt1.negate();

System.out.println("Absolute Value: " + absValue);

System.out.println("Negative Value: " + negValue);

}}

高精度：转化为字符串