# 13、罗马数字转整数（Roman to Integer）

## 题目

罗马数字包含以下七种字符: I， V， X， L，C，D 和 M。

**字符** **数值**  
I 1  
V 5  
X 10  
L 50  
C 100  
D 500  
M 1000  
例如， 罗马数字 2 写做 II ，即为两个并列的 1。12 写做 XII ，即为 X + II 。 27 写做 XXVII, 即为 XX + V + II 。

通常情况下，罗马数字中小的数字在大的数字的右边。但也存在特例，例如 4 不写做 IIII，而是 IV。数字 1 在数字 5 的左边，所表示的数等于大数 5 减小数 1 得到的数值 4 。同样地，数字 9 表示为 IX。这个特殊的规则只适用于以下六种情况：

I 可以放在 V (5) 和 X (10) 的左边，来表示 4 和 9。  
X 可以放在 L (50) 和 C (100) 的左边，来表示 40 和 90。   
C 可以放在 D (500) 和 M (1000) 的左边，来表示 400 和 900。  
给定一个罗马数字，将其转换成整数。输入确保在 1 到 3999 的范围内。

**示例 1:**

输入: "III"  
输出: 3

**示例 2:**

输入: "IV"  
输出: 4

**示例 3:**

输入: "IX"  
输出: 9

**示例 4:**

输入: "LVIII"  
输出: 58  
解释: L = 50, V= 5, III = 3.

**示例 5:**

输入: "MCMXCIV"  
输出: 1994  
解释: M = 1000, CM = 900, XC = 90, IV = 4.

## 解答：

* 判断赋值

|  |
| --- |
| //法一  public static int RomanToInt(string s)  {  //注意观察本题  //例如Ⅳ这个数字是4，但是如果按顺序加起来是6.这两个数字相差2  //其他数字也是类似，Ⅸ也是差2  //在例如XL按顺序是60，但是实际表示为40，以此类推  int sum = 0;  //检测指定字符串从零开始的索引  //如果找到该字符串，则为 value 的从零开始的索引位置；  //如果未找到该字符串，则为 -1。  //如果 value 为 String.Empty，则返回值为 0。  if (s.IndexOf("IV") != -1) { sum -= 2; }//这样写短  if (s.IndexOf("IX") != -1) { sum -= 2; }  if (s.IndexOf("XL") != -1) { sum -= 20; }  if (s.IndexOf("XC") != -1) { sum -= 20; }  if (s.IndexOf("CD") != -1) { sum -= 200; }  if (s.IndexOf("CM") != -1) { sum -= 200; }  for (int i = 0; i < s.Length; i++)  {  //指定位置处的元素Substring  char c = s.ElementAt(i);  if (c == 'I') { sum += 1; }  if (c == 'V') { sum += 5; }  if (c == 'X') { sum += 10; }  if (c == 'L') { sum += 50; }  if (c == 'C') { sum += 100; }  if (c == 'D') { sum += 500; }  if (c == 'M') { sum += 1000; }  }  return sum;  } |

* 利用字典

|  |
| --- |
| //法二  public static int RomanToInt1(string s)  {  //也可以利用字典  //先把这些字母的值放在字典中，然后把传进来的字符串根据字典的值加在最后的结果中  //注意：  //1、没有数字的收获时候输出0  //2、最后一位必定要加在结果上  //3、输入的字符串的前一位比后一位大时，把字典中对应的值加给结果  //4、输入的字符串的前一位比后一位小时，把字典中对应的值减给结果  Dictionary<char, int> RomanNumeralMap = new Dictionary<char, int>();  RomanNumeralMap['I'] = 1;  RomanNumeralMap['V'] = 5;  RomanNumeralMap['X'] = 10;  RomanNumeralMap['L'] = 50;  RomanNumeralMap['C'] = 100;  RomanNumeralMap['D'] = 500;  RomanNumeralMap['M'] = 1000;  int sum = 0;  if (s.Length == 0)  {  return 0;  }  //最后一位直接加  sum += RomanNumeralMap[s[s.Length - 1]];  //只要找到倒数第二位  for (int i = 0; i < s.Length - 1; i++)  {  //字符串的前一位比后一位大时，把字典中对应的值加给结果  //字符串实际上也是字符组成的数组  if (RomanNumeralMap[s[i]] >= RomanNumeralMap[s[i + 1]])  {  sum += RomanNumeralMap[s[i]];  }  //字符串的前一位比后一位小时，把字典中对应的值减给结果  else  {  sum -= RomanNumeralMap[s[i]];  }  }  return sum;  } |

* 测试

|  |
| --- |
| static void Main(string[] args)  {  string[] s = new string[] { "III", "IV", "IX", "LVIII", "MCMXCIV" };  for (int i = 0; i < s.Length; i++)  {  int n = RomanToInt(s[i]);  Console.WriteLine("罗马数字：{0}\n阿拉伯数字：{1}", s[i], n);  }  for (int i = 0; i < s.Length; i++)  {  int n1 = RomanToInt1(s[i]);  Console.WriteLine("罗马数字：{0}\n阿拉伯数字：{1}", s[i], n1);  }  Console.ReadKey();  } |