我的代码

using System;

public delegate bool TwoIntsPredicate(int n1, int n2);

class Delegates

{

static void Main(string[] args)

{

Random rd = new Random();

int[] numbers = new int[15];

for (int i = 0; i < numbers.Length; i++)

numbers[i] = rd.Next(-100, 100);

Console.Write("原始数组：");

Array.ForEach(numbers, x => Console.Write($"{x} "));

Console.WriteLine();

//生成委托实例

TwoIntsPredicate gt = GreaterThan;

//对数值15，利用委托变量来判断该值是否为偶数

// ————a—————

Console.WriteLine($"以5、10为参数，调用委托变量gt: {gt(5, 10)}");

//利用委托，选出最大值

int maxValue = ExtremeValue(numbers, GreaterThan);

Console.WriteLine($"numbers数组中的最大值是{maxValue}");

//选出最大值的另一种写法：利用Lambda表达式

TwoIntsPredicate gt2 = (x, y) => x > y;

maxValue = ExtremeValue(numbers, gt2);

Console.WriteLine($"numbers数组中的最大值是{maxValue}");

//模仿上面方法，写出选出最小值的两种写法

//方法一：利用后文定义的函数

// ————b—————

TwoIntsPredicate lt = LesserThan;

int minValue = ExtremeValue(numbers, lt);

Console.WriteLine($"numbers数组中的最小值是{minValue}");

//方法二：利用Lambda表达式

// ————c—————

TwoIntsPredicate lt2 = (x, y) => x < y;

minValue = ExtremeValue(numbers, lt2);

Console.WriteLine($"numbers数组中的最小值是{minValue}");

//模仿选出最大最小值等操作，请自行用lambda表达式创建一个委托

//找出绝对值最大的数，例如 -99比88大

// ————d—————

TwoIntsPredicate absMax = (x, y) => Math.Abs(x) > Math.Abs(y);

int absMaxValue = ExtremeValue(numbers, absMax);

Console.WriteLine($"numbers数组中绝对值最大的数是{absMaxValue}");

//模仿上述操作，请自行创建一个函数DigitSumGreaterThan，

//它将数值各数位和相加再进行比较，如79数位之和为16，比92的数位之和11大，

//利用它找出数组中的极值

// ————e—————

TwoIntsPredicate digitSumGreaterThan = (x, y) => DigitSumGreaterThan(x, y);

int digitSumMaxValue = ExtremeValue(numbers, digitSumGreaterThan);

Console.WriteLine($"numbers数组中各数位和最大的数是{digitSumMaxValue}");

}

//判断是否大于

private static bool GreaterThan(int n1, int n2)

{

return n1 > n2;

}

//判断是否小于

private static bool LesserThan(int n1, int n2)

{

return n1 < n2;

}

//计算数值的各数位之和

private static bool DigitSumGreaterThan(int n1, int n2)

{

// ————f—————

int digitSumN1 = 0;

int digitSumN2 = 0;

// 计算n1的各位数之和

int temp = n1;

while (temp != 0)

{

digitSumN1 += temp % 10;

temp /= 10;

}

// 计算n2的各位数之和

temp = n2;

while (temp != 0)

{

digitSumN2 += temp % 10;

temp /= 10;

}

// 返回各位数之和比较结果

return digitSumN1 > digitSumN2;

}

//从array数组中，根据predicate指定的规则，找出极值

private static int ExtremeValue(int[] array, TwoIntsPredicate predicate)

{

// ————g—————

int extremeValue = array[0];

foreach (var num in array)

{

if (predicate(num, extremeValue))

{

extremeValue = num;

}

}

return extremeValue;

}

}

代码截图

