**Тестовые задания**

1. **Ломай меня полностью.**

Реализуйте метод FailProcess так, чтобы процесс завершался. Предложите побольше различных решений.

using System;

class Program

{

static void Main(string[] args)

{

try

{

FailProcess();

}

catch { }

Console.WriteLine("Failed to fail process!");

Console.ReadKey();

}

static void FailProcess(){ //... write your code here }

}

1. **Операция «Ы».**

Что выводится на экран? Измените класс Number так, чтобы на экран выводился результат сложения для любых значений someValue1 и someValue2.

using System;

using System.Globalization;

class Program

{

static readonly IFormatProvider \_ifp = CultureInfo.InvariantCulture;

class Number

{

readonly int \_number;

public Number(int number)

{

\_number = number;

}

public override string ToString()

{

return \_number.ToString(\_ifp);

}

}

static void Main(string[] args)

{

int someValue1 = 10;

int someValue2 = 5;

string result = new Number(someValue1) + someValue2.ToString(\_ifp);

Console.WriteLine(result);

Console.ReadKey();

}

}

1. **Мне только спросить!**Реализуйте метод по следующей сигнатуре:

/// <summary>  
/// <para> Отсчитать несколько элементов с конца </para>  
/// <example> new[] {1,2,3,4}.EnumerateFromTail(2) = (1, ), (2, ), (3, 1), (4, 0)</example>  
/// </summary>   
/// <typeparam name="T"></typeparam>  
/// <param name="enumerable"></param>  
/// <param name="tailLength">Сколько элеметнов отсчитать с конца  (у последнего элемента tail = 0)</param>  
/// <returns></returns>  
public static IEnumerable<(T item, int? tail)> EnumerateFromTail<T>(this IEnumerable<T> enumerable, int? tailLength)  
  
*Возможно ли реализовать такой метод выполняя перебор значений перечисления только 1 раз?*

1. **Высший сорт.**

Реализуйте метод Sort. Известно, что потребители метода зачастую не будут вычитывать данные до конца. Оптимально ли Ваше решение с точки зрения скорости выполнения? С точки зрения потребляемой памяти?

/// <summary>

/// Возвращает отсортированный по возрастанию поток чисел

/// </summary>

/// <param name="inputStream">Поток чисел от 0 до maxValue. Длина потока не превышает миллиарда чисел.</param>

/// <param name="sortFactor">Фактор упорядоченности потока. Неотрицательное число. Если в потоке встретилось число x, то в нём больше не встретятся числа меньше, чем (x - sortFactor).</param>

/// <param name="maxValue">Максимально возможное значение чисел в потоке. Неотрицательное число, не превышающее 2000.</param>

/// <returns>Отсортированный по возрастанию поток чисел.</returns>

IEnumerable<int> Sort(IEnumerable<int> inputStream, int sortFactor, int maxValue)

1. **Слон из мухи.**

Программа выводит на экран строку «Муха», а затем продолжает выполнять остальной код. Реализуйте метод TransformToElephant так, чтобы программа выводила на экран строку «Слон», а затем продолжала выполнять остальной код, не выводя перед этим на экран строку «Муха».

using System;

class Program

{

static void Main(string[] args)

{

TransformToElephant();

Console.WriteLine("Муха");

//... custom application code

}

static void TransformToElephant() { //... write your code here }

}