Задание.

Вывести массив из N целых положительных чисел четырёхзначных чисел. Вывести числа, у которых максимальная цифра 7, и среднее арифметическое цифр не более 7,если таких чисел нет, то вывести сообщение об этом.

Создаем Модульные тесты:

Module1 – Проверка количества символов в строке , подсчет количества цифр в числе, проверка на положительность чисел

Module2 – Проверка нахождения максимальной цифры в числе и равна ли она 7, проверка средне арифметического, вывод таких чисел.

Создание UI тестов:

CodedUITestMethod1() – Первый тест

this.UIMap.RecordedMethod1() - открыть приложение

this.UIMap.Second\_Test() - проверка второго теста на вывод нужных значений

this.UIMap.Close() - закрытие приложения

CodedUITestMethod3() – Второй тест

this.UIMap.RecordedMethod2() - отрыть и выполнить действие

this.UIMap.TEst\_3(); - проверка есть ли такие числа и проверка что таких нет.

this.UIMap.Close() - закрыть

**План Тестирования**

|  |  |
| --- | --- |
| Вход | Выход |
| Module1.CheckCountDigit("1004") | true |
| Module2.NumberWithOptions("1007 2007 1001"), | 1007 |
| Module2.CheckSevenAverage("1007"), | true |

В результате должны получиться следующие методы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

// Вывести массив из 10 целых положительных чисел Вывести количество чисел массива , у которых максимальная цивра равна 7

//и средне арифмитическое этих чисел меньше 7, если таких элементов нет то вывести сообщения об этом

namespace Lab3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

String text = textBox2.Text;

//MessageBox.Show(text);

String result = "";

if (Module1.CheckCountDigit(text) && Module1.PositiveDigits(text))

{

result = Module2.NumberWithOptions(text);

// MessageBox.Show(Module2.CheckSeven(text).ToString());

// MessageBox.Show(Module2.CheckSevenAverage(text).ToString());

if (result != "")

{

textBox1.Text = result;

}

else

{

MessageBox.Show("Таких чисел нет");

//Module2.NumberWithOptions(text);

}

}

else

{

MessageBox.Show("Неправильно введена строка нужно чтобы было 10 положительных цифр введнных через пробел");

// MessageBox.Show(Module1.CheckCountDigit(text).ToString());

//MessageBox.Show(Module1.checkStrLength(text).ToString());

//MessageBox.Show(Module1.PositiveDigits(text).ToString());

}

}

public class Module2

{

public static Boolean CheckSeven(String s) // Подсчет слов , у которых нечетное количество сиволов и если их больше 3 то тру

{

//Char delimiter = ' ';

//String[] substrings = s.Split(delimiter);// делим строку по пробелам

int max = 0;

Char[] t = s.ToCharArray(0, s.Length);

foreach(var i in t)

{

int number = int.Parse(i.ToString());

if ( max < number)

max = number;

}

//MessageBox.Show(max.ToString());

if (max == 7)

{

return true;

}

return false;

}

public static void CheckSeven1(String s) // Подсчет слов , у которых нечетное количество сиволов и если их больше 3 то тру

{

//Char delimiter = ' ';

//String[] substrings = s.Split(delimiter);// делим строку по пробелам

int max = 0;

Char[] t = s.ToCharArray(0, s.Length);

foreach (var i in t)

{

int number = int.Parse(i.ToString());

//MessageBox.Show(number.ToString());

if (max < number)

max = number;

}

// MessageBox.Show(max.ToString());

}

public static Boolean CheckSevenAverage(String s) // Подсчет слов , у которых нечетное количество сиволов и если их больше 3 то тру

{

int sum = 0;

foreach (var t in s)

{

int number = int.Parse(t.ToString());

sum += number;

}

sum = sum / 4;

if (sum < 7)

{

return true;

}

return false;

}

public static String NumberWithOptions(String s) // Подсчет слов , у которых нечетное количество сиволов и если их больше 3 то тру

{

Boolean result = false;

Char delimiter = ' ';

String[] substrings = s.Split(delimiter);// делим строку по пробелам

String str = "";

int sum = 0;

foreach (var t in substrings)

{

if (CheckSeven(t) && CheckSevenAverage(t))

{

//MessageBox.Show(CheckSeven(t).ToString());

//MessageBox.Show(CheckSevenAverage(t).ToString());

str +=" " + t;

}

}

return str;

}

}

public class Module1 // Модуль один на проверку количества символов в строке, строки должны быть меньше 30,

{

public static Boolean checkStrLength(String s)

{

if (s.Length < 55)

return true;

else

return false;

}

public static Boolean CheckCountDigit(String s) // Подсчет слов , у которых нечетное количество сиволов и если их больше 3 то тру

{

Boolean result = false;

Char delimiter = ' ';

String[] substrings = s.Split(delimiter);// делим строку по пробелам

// MessageBox.Show(substrings.ToString());

foreach (var t in substrings)

{

// MessageBox.Show(t);

if (t.Length != 4)

return false;

}

return true;

}

public static Boolean PositiveDigits(String s) // Положительные ли числа

{

Char delimiter = ' ';

String[] substrings = s.Split(delimiter);// делим строку по пробелам

foreach (var t in substrings)

{

int number = int.Parse(t);

// MessageBox.Show(number.ToString());

if (number < 0)

return false;

}

return true;

}

// Написать программу по определнию есть ли сиволы в строке или удаление символов в строке

}

private void button2\_Click(object sender, EventArgs e)

{

String lol = textBox1.Text;

Module2.CheckSeven1(lol);

}

}

}

Код Тестого модуля

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Lab3;

namespace Modul1

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

{

// Проверка количества чисел

Assert.AreEqual(Form1.Module1.CheckCountDigit("1004"),true);

Assert.AreEqual(Form1.Module1.CheckCountDigit("100"), false);

// Проверка позитивных чисел

Assert.AreEqual(Form1.Module1.PositiveDigits("3423 3423 4353"), true);

Assert.AreEqual(Form1.Module1.PositiveDigits("3423 3423 -4353"), false);

// Проверка 7 - как самая большое число

Assert.AreEqual(Form1.Module2.CheckSeven("1007"),true);

Assert.AreEqual(Form1.Module2.CheckSeven("1078"), false);

// Проверка средне арифмитическое меньше 7

Assert.AreEqual(Form1.Module2.CheckSevenAverage("1007"), true);

Assert.AreEqual(Form1.Module2.CheckSevenAverage("9999"), false);

// Проверка всех значений в списке

Assert.AreEqual(Form1.Module2.NumberWithOptions("1007 2007 1001"), " 1007 2007");

Assert.AreEqual(Form1.Module2.NumberWithOptions("1001 2001 1001"), "");

}

}

}

Код UI теста:

using System;

using System.Collections.Generic;

using System.Text.RegularExpressions;

using System.Windows.Input;

using System.Windows.Forms;

using System.Drawing;

using Microsoft.VisualStudio.TestTools.UITesting;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Microsoft.VisualStudio.TestTools.UITest.Extension;

using Keyboard = Microsoft.VisualStudio.TestTools.UITesting.Keyboard;

namespace CodedUITestProject1

{

/// <summary>

/// Сводное описание для CodedUITest1

/// </summary>

[CodedUITest]

public class CodedUITest1

{

public CodedUITest1()

{

}

[TestMethod]

public void CodedUITestMethod2()

{

this.UIMap.RecordedMethod2(); // октрыть

this.UIMap.AssertMethod1(); // проверка 2

this.UIMap.Close(); // закрыть

// Чтобы создать код для этого теста, выберите в контекстном меню команду "Формирование кода для кодированного теста пользовательского интерфейса", а затем выберите один из пунктов меню.

}

public void CodedUITestMethod1()

{

this.UIMap.RecordedMethod1(); // октрыть

this.UIMap.Second\_Test(); // проверка 1

this.UIMap.Close(); // закрыть

// Чтобы создать код для этого теста, выберите в контекстном меню команду "Формирование кода для кодированного теста пользовательского интерфейса", а затем выберите один из пунктов меню.

}

public void CodedUITestMethod3()

{

this.UIMap.RecordedMethod2();// отрыть и выполнить действие

this.UIMap.TEst\_3(); // Test 3

this.UIMap.Close(); // закрыть

// Чтобы создать код для этого теста, выберите в контекстном меню команду "Формирование кода для кодированного теста пользовательского интерфейса", а затем выберите один из пунктов меню.

}

#region Дополнительные атрибуты тестирования

// При написании тестов можно использовать следующие дополнительные атрибуты:

////TestInitialize используется для выполнения кода перед запуском каждого теста

//[TestInitialize()]

//public void MyTestInitialize()

//{

// // Чтобы создать код для этого теста, выберите в контекстном меню команду "Формирование кода для кодированного теста пользовательского интерфейса", а затем выберите один из пунктов меню.

//}

////TestCleanup используется для выполнения кода после завершения каждого теста

//[TestCleanup()]

//public void MyTestCleanup()

//{

// // Чтобы создать код для этого теста, выберите в контекстном меню команду "Формирование кода для кодированного теста пользовательского интерфейса", а затем выберите один из пунктов меню.

//}

#endregion

/// <summary>

///Получает или устанавливает контекст теста, в котором предоставляются

///сведения о текущем тестовом запуске и обеспечивается его функциональность.

///</summary>

public TestContext TestContext

{

get

{

return testContextInstance;

}

set

{

testContextInstance = value;

}

}

private TestContext testContextInstance;

public UIMap UIMap

{

get

{

if ((this.map == null))

{

this.map = new UIMap();

}

return this.map;

}

}

private UIMap map;

}

}