**Отчет по ПР9**

**Лопаткин Сергей ИСП-41**

Структура взаимодействия модулей выглядит следующим образом, показанным на рисунке 1.

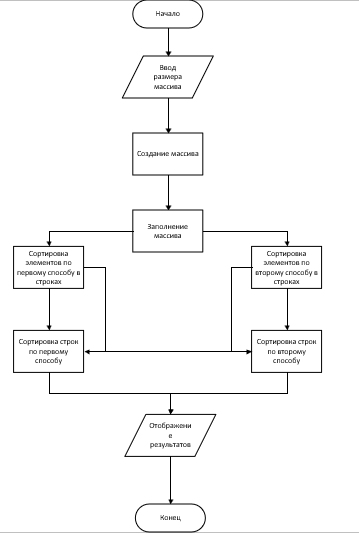
****

Рисунок 1 – Структура взаимодействия модулей

Используется отдельная библиотека .dll, которая несет в себе создание массива и его сортировку. Диаграмма классов его выглядит следующим образом как на рисунке 2.

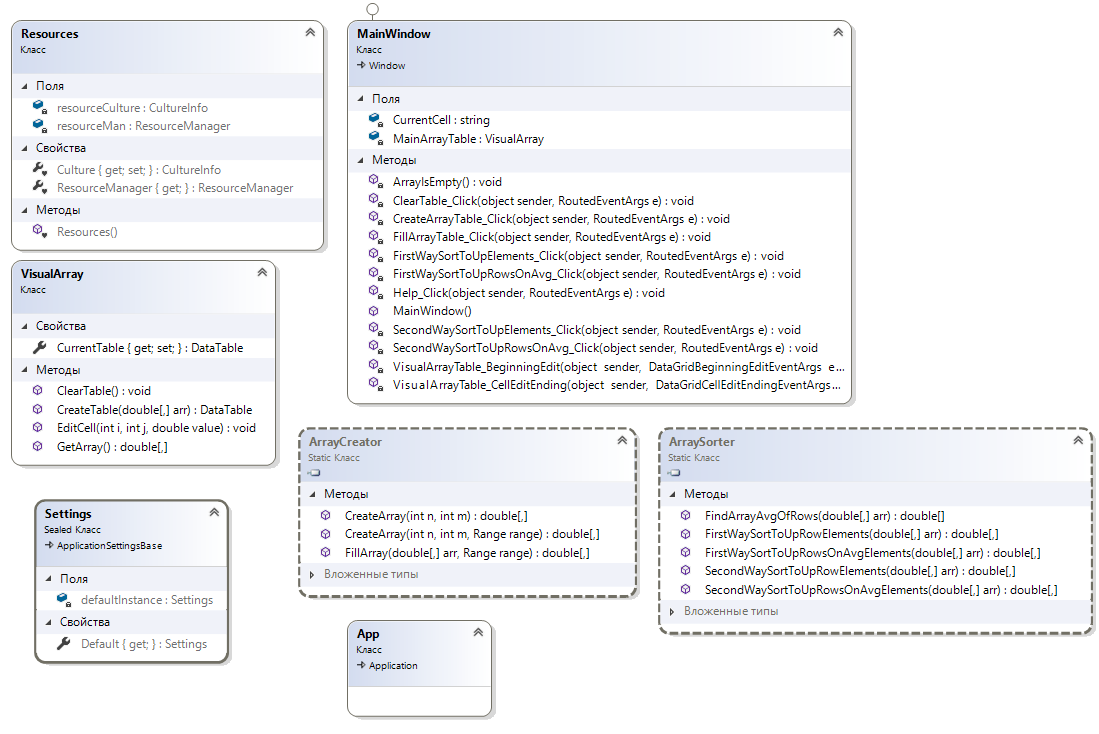
****

Рисунок 2 – Диаграмма классов

Интегрирование модулей было постепенным (последовательным). Сначала тестировалось создание массива, заполненного нулями. Позже происходило тестирование создания заполненного массива. В дальнейшем, без юнит-теста, был добавлен простой заполнитель массива (код метода лежит на основе «создать+заполнить», только без создания пустого массива, а с заполнением текущего). После этого производилась проверка правильной сортировки массива. Сначала способы сортировки элементов, а позднее – строк. Структурная схема взаимодействия модулей показывать также процесс тестирования компонентов.

После утверждения верности алгоритмов, создавался интерфейс. Некоторые методы (для минимизации размеров кода в главном коде программы) были прописаны в классы, необходимые для автоматизации процесса взаимодействия (в частности, использовалась модернизация класса VisualArray, который был написан с нуля).

Если смотреть на структурную схему взаимодействия и учесть факт тестирования, соответствующий ее последовательности, то у нас использовалось нисходящее интеграционное тестирование.

После всех успехов, дополнительно, произведена была очистка кода.

Также, если коротко пояснить значение фазы интеграции модулей, то нужно постепенно интегрировать, чтобы выявлять проблемы и исправлять их по мере поступления, проходя этапы. Это позволит в ускоренном темпе выявить недочеты, допущенные при проектировании, что сэкономит достаточный промежуток времени на поиски «неизвестного». Существует несколько подходов:

* Функционально-ориентированная интеграция

Еще один поход — интеграция одной функции в каждый момент времени. Под «функцией» понимается не нечто расплывчатое, а какое-нибудь поддающееся определению свойство системы, в которой выполняется интеграция. Когда интегрируемая функция превышает по размерам отдельный класс, то «единица приращения» инкрементной интеграции становится больше отдельного класса. Это немного снижает преимущество инкрементного подхода в том плане, что уменьшает вашу уверенность об источнике новых ошибок. Однако если вы тщательно тестировали классы, реализующие эту функцию, перед интеграцией, то это лишь небольшой недостаток. Вы можете использовать стратегии инкрементной интеграции рекурсивно, сформировав сначала из небольших кусков отдельные свойства, а затем инкрементно объединив их в систему. Обычно процесс начинается с формирования скелета, поскольку он способен поддерживать остальную функциональность. В интерактивной системе такой изначальной опцией может стать система интерактивного меню. Вы можете прикреплять остальную функциональность к той опции, которую интегрировали первой.

* Риск-ориентированная интеграция

При риск-ориентированной интеграции вы определяете степень риска, связанную с каждым классом. Вы решаете, какие части системы будут самыми трудными, и реализуете их первыми.

* Сэндвич-интеграция

Проблемы с нисходящей и восходящей интеграциями в чистом виде привели к тому, что некоторые эксперты стали рекомендовать сэндвич-подход. Сначала вы объединяете высокоуровневые классы бизнес-объектов на вершине иерархии. Затем добавляете классы, взаимодействующие с аппаратной частью, и широко используемые вспомогательные классы в низу иерархии. Напоследок вы оставляете классы среднего уровня.

* Восходящая интеграция

При восходящей интеграции вы пишете и интегрируете сначала классы, находящиеся внизу иерархии.

* Нисходящая интеграция

При нисходящей интеграции класс на вершине иерархии пишется и интегрируется первым.

* Т-образная интеграция

Последний подход, который часто упоминается в связи с проблемами нисходящей и восходящей методик, называется «Т-образной интеграцией». При таком подходе выбирается некоторый вертикальный слой, который разрабатывается и интегрируется раньше других. Этот слой должен проходить сквозь всю систему от начала до конца и позволять выявлять основные проблемы в допущениях, сделанных при проектировании системы. Реализовав этот вертикальный участок (и устранив все связанные с этим проблемы), можно разрабатывать основную канву системы (например, системное меню для настольного приложения).

Мои предпочтения для сложных проектов – функционально-ориентированная интеграция. Для простых – нисходящая. В текущем проекте использовалась основа: положи фундамент-построй здание-модернизируй. (используется в крылатом понимании данная основа). Основа выбора интеграции – как лучше?; как бы не запутаться?;что удобнее?;

**Листинг главного кода программы**

using CreatingArray;

using System;

using System.Windows;

using System.Windows.Controls;

namespace PW9\_02\_02\_ArrayDifSort

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

VisualArray MainArrayTable = new VisualArray();

private void CreateArrayTable\_Click(object sender, RoutedEventArgs e)

{

try

{

VisualArrayTable.ItemsSource = MainArrayTable.CreateTable(ArrayCreator.CreateArray(Convert.ToInt32(RowCount.Text), Convert.ToInt32(ColumnCount.Text))).DefaultView;

TableTab.Focus();

}

catch

{

MessageBox.Show("Некорректно введены значения строк или столбцов!", "Некорректность данных", MessageBoxButton.OK, MessageBoxImage.Error);

ControlTab.Focus();

}

}

private void FillArrayTable\_Click(object sender, RoutedEventArgs e)

{

try

{

if (VisualArrayTable.ItemsSource == null) throw new Exception();

VisualArrayTable.ItemsSource = MainArrayTable.CreateTable(ArrayCreator.FillArray(MainArrayTable.GetArray(), new ArrayCreator.Range(Convert.ToInt32(FirstValue.Text), Convert.ToInt32(SecondValue.Text)))).DefaultView;

TableTab.Focus();

}

catch

{

MessageBox.Show("Некорректно введены значения диапазона или массива не существует!", "Ошибка данных", MessageBoxButton.OK, MessageBoxImage.Error);

ControlTab.Focus();

}

}

private void ClearTable\_Click(object sender, RoutedEventArgs e)

{

MainArrayTable.ClearTable();

VisualArrayTable.ItemsSource = null;

TableTab.Focus();

}

private void FirstWaySortToUpElements\_Click(object sender, RoutedEventArgs e)

{

try

{

VisualArrayTable.ItemsSource = MainArrayTable.CreateTable(ArraySorter.FirstWaySortToUpRowElements(MainArrayTable.GetArray())).DefaultView;

TableTab.Focus();

}

catch

{

ArrayIsEmpty();

}

}

private void SecondWaySortToUpElements\_Click(object sender, RoutedEventArgs e)

{

try

{

VisualArrayTable.ItemsSource = MainArrayTable.CreateTable(ArraySorter.SecondWaySortToUpRowElements(MainArrayTable.GetArray())).DefaultView;

TableTab.Focus();

}

catch

{

ArrayIsEmpty();

}

}

private void FirstWaySortToUpRowsOnAvg\_Click(object sender, RoutedEventArgs e)

{

try

{

VisualArrayTable.ItemsSource = MainArrayTable.CreateTable(ArraySorter.FirstWaySortToUpRowsOnAvgElements(MainArrayTable.GetArray())).DefaultView;

TableTab.Focus();

}

catch

{

ArrayIsEmpty();

}

}

private void SecondWaySortToUpRowsOnAvg\_Click(object sender, RoutedEventArgs e)

{

try

{

VisualArrayTable.ItemsSource = MainArrayTable.CreateTable(ArraySorter.SecondWaySortToUpRowsOnAvgElements(MainArrayTable.GetArray())).DefaultView;

TableTab.Focus();

}

catch

{

ArrayIsEmpty();

}

}

private void ArrayIsEmpty()

{

MessageBox.Show("Отсутствует массив для сортировки данных!", "Сортировка данных", MessageBoxButton.OK, MessageBoxImage.Error);

ControlTab.Focus();

}

private void Help\_Click(object sender, RoutedEventArgs e)

{

MessageBox.Show("Разработчик: Лопаткин Сергей ИСП-41\n" +

"GitHub.Name=HaproBishop\n" +

"Задача. Задан двумерный массив размерности n x m. " +

"Отсортировать элементы строк массива по возрастанию значений, а затем отсортировать строки массива по возрастанию среднего " +

"арифметического элементов строк. " +

"Реализовать сортировку разными способами и сравнить эффективность этих способов для разных исходных данных.",

"Справка", MessageBoxButton.OK, MessageBoxImage.Information);

}

string CurrentCell = "";

private void VisualArrayTable\_BeginningEdit(object sender, System.Windows.Controls.DataGridBeginningEditEventArgs e)

{

int iRow = e.Row.GetIndex();

int iColumn = e.Column.DisplayIndex;

CurrentCell = MainArrayTable.GetArray()[iRow, iColumn].ToString();

}

private void VisualArrayTable\_CellEditEnding(object sender, System.Windows.Controls.DataGridCellEditEndingEventArgs e)

{

if (!double.TryParse(((TextBox)e.EditingElement).Text, out \_))

{

((TextBox)e.EditingElement).Text = CurrentCell;

e.Cancel = true;

}

}

}

}

**Листинг окна программы**

<Window x:Class="PW9\_02\_02\_ArrayDifSort.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:PW9\_02\_02\_ArrayDifSort"

mc:Ignorable="d"

Title="Сортировка массива" MinHeight="450" MinWidth="800" WindowStartupLocation="CenterScreen" FontSize="14">

<DockPanel>

<Menu DockPanel.Dock="Top">

<MenuItem x:Name="CreateArrayTable" Header="Создать" Click="CreateArrayTable\_Click"/>

<MenuItem x:Name="FillArrayTable" Header="Заполнить" Click="FillArrayTable\_Click"/>

<MenuItem x:Name="ClearTable" Header="Очистить" Click="ClearTable\_Click"/>

<MenuItem Header="Сортировка (1 способ)">

<MenuItem x:Name="FirstWaySortToUpElements" Header="Элементы по возрастанию" ToolTip="Отсортировать элементы строк массива по возрастанию их значений" Click="FirstWaySortToUpElements\_Click"/>

<MenuItem x:Name="FirstWaySortToUpRowsOnAvg" Header="Строки по возрастанию среднего" ToolTip="Отсортировать строки массива по возрастанию среднего арифметического элементов строк" Click="FirstWaySortToUpRowsOnAvg\_Click"/>

</MenuItem>

<MenuItem Header="Сортировка (2 способ)">

<MenuItem x:Name="SecondWaySortToUpElements" Header="Элементы по возрастанию" ToolTip="Отсортировать элементы строк массива по возрастанию их значений" Click="SecondWaySortToUpElements\_Click"/>

<MenuItem x:Name="SecondWaySortToUpRowsOnAvg" Header="Строки по возрастанию среднего" ToolTip="Отсортировать строки массива по возрастанию среднего арифметического элементов строк" Click="SecondWaySortToUpRowsOnAvg\_Click"/>

</MenuItem>

<MenuItem x:Name="Help" Header="Справка" Click="Help\_Click"/>

</Menu>

<TabControl>

<TabItem x:Name="TableTab" Header="Таблица">

<GroupBox Header="Двумерный массив">

<DataGrid x:Name="VisualArrayTable" CanUserAddRows="False" BeginningEdit="VisualArrayTable\_BeginningEdit" CellEditEnding="VisualArrayTable\_CellEditEnding"/>

</GroupBox>

</TabItem>

<TabItem x:Name="ControlTab" Header="Управление">

<StackPanel>

<GroupBox Header="Создание">

<StackPanel>

<Label Content="Строки:"/>

<TextBox x:Name="RowCount" Margin="3" MaxLength="2"/>

<Label Content="Столбцы:"/>

<TextBox x:Name="ColumnCount" Margin="3" MaxLength="2"/>

</StackPanel>

</GroupBox>

<GroupBox Header="Заполнение">

<StackPanel>

<Label Content="Диапазон"/>

<Separator/>

<Label Content="от:"/>

<TextBox x:Name="FirstValue" Margin="3" MaxLength="5"/>

<Label Content="до:"/>

<TextBox x:Name="SecondValue" Margin="3" MaxLength="5"/>

</StackPanel>

</GroupBox>

</StackPanel>

</TabItem>

</TabControl>

</DockPanel>

</Window>

**Листинг класса ArrayCreator**

using System;

namespace CreatingArray

{

public static class ArrayCreator

{

public struct Range

{

private int \_secondValue;

public Range(int firstValue, int secondValue)

{

FirstValue = firstValue;

\_secondValue = secondValue;

}

public int FirstValue { get; set; }

public int SecondValue

{

get => \_secondValue;

set

{

if (FirstValue > value) throw new Exception("Второе значение диапазона больше или равно первому!");

\_secondValue = value;

}

}

}

/// <summary>

/// Создает двумерный массив данных, заполненный нулями.

/// </summary>

/// <param name="n"></param>

/// <param name="m"></param>

/// <returns></returns>

public static double[,] CreateArray(int n, int m)

{

double[,] arr = new double[n, m];

return arr;

}

public static double[,] CreateArray(int n, int m, Range range)

{

var arr = CreateArray(n, m);

Random rnd = new Random();

for (int i = 0; i < arr.GetLength(0); i++)

{

for (int j = 0; j < arr.GetLength(1); j++)

{

arr[i, j] = rnd.Next(range.FirstValue, range.SecondValue);

}

}

return arr;

}

public static double[,] FillArray(double[,] arr, Range range)

{

Random rnd = new Random();

for (int i = 0; i < arr.GetLength(0); i++)

{

for (int j = 0; j < arr.GetLength(1); j++)

{

arr[i, j] = rnd.Next(range.FirstValue, range.SecondValue);

}

}

return arr;

}

}

}

**Листинг класса ArraySorter**

namespace CreatingArray

{

public static class ArraySorter

{

public struct TemporaryValue

{

public double Value { get; set; }

public int FirstIndex { get; set; }

public int SecondIndex { get; set; }

public TemporaryValue(int firstIndex, int secondIndex, double value)

{

FirstIndex = firstIndex;

SecondIndex = secondIndex;

Value = value;

}

}

//Правился при тестировании

public static double[,] FirstWaySortToUpRowElements(double[,] arr)

{

for (int i = 0; i < arr.GetLength(0); i++)

{

for (int jT = 0; jT < arr.GetLength(1) - 1; jT++)

{

TemporaryValue temporaryValue = new TemporaryValue(i, jT, arr[i, jT]);

for (int j = jT + 1; j < arr.GetLength(1); j++)

{

if (arr[i, jT] > arr[i, j])

{

temporaryValue.Value = arr[i, j];

arr[i, j] = arr[temporaryValue.FirstIndex, temporaryValue.SecondIndex];

arr[temporaryValue.FirstIndex, temporaryValue.SecondIndex] = temporaryValue.Value;

}

}

}

}

return arr;

}

//Исправлялся при тестировании

public static double[,] SecondWaySortToUpRowElements(double[,] arr)

{

for (int i = 0; i < arr.GetLength(0); i++)

{

for (int jT = 0; jT < arr.GetLength(1) - 1; jT++)

{

for (int j = jT + 1; j < arr.GetLength(1); j++)

{

if (arr[i, jT] > arr[i, j])

{

double temporaryValue = arr[i, j];

arr[i, j] = arr[i, jT];

arr[i, jT] = temporaryValue;

}

}

}

}

return arr;

}

public static double[] FindArrayAvgOfRows(double[,] arr)

{

double[] avgArr = new double[arr.GetLength(0)];

for (int i = 0; i < arr.GetLength(0); i++)

{

double avg = 0;

for (int j = 0; j < arr.GetLength(1); j++)

{

avg += arr[i, j];

}

avgArr[i] = avg / arr.GetLength(1);

}

return avgArr;

}

//Исправлялся при тестировании

public static double[,] FirstWaySortToUpRowsOnAvgElements(double[,] arr)

{

double[] avgArr = FindArrayAvgOfRows(arr);

double tempVal;

for (int iT = 0; iT < arr.GetLength(0) - 1; iT++)

{

TemporaryValue temporaryValue = new TemporaryValue(iT, 0, avgArr[iT]);

for (int i = iT + 1; i < arr.GetLength(0); i++)

{

if (temporaryValue.Value > avgArr[i])

{

for (int j = 0; j < avgArr.Length; j++)

{

tempVal = arr[i, j];

arr[i, j] = arr[iT, j];

arr[iT, j] = tempVal;

}

temporaryValue.Value = avgArr[i];

tempVal = avgArr[iT];

avgArr[iT] = avgArr[i];

avgArr[i] = tempVal;

}

}

}

return arr;

}

//Исправлялся при тестировании

public static double[,] SecondWaySortToUpRowsOnAvgElements(double[,] arr)

{

double[] avgArr = FindArrayAvgOfRows(arr);

double tempVal;

for (int iT = 0; iT < arr.GetLength(0) - 1; iT++)

{

for (int i = iT + 1; i < arr.GetLength(0); i++)

{

if (avgArr[iT] > avgArr[i])

{

for (int j = 0; j < avgArr.Length; j++)

{

tempVal = arr[i, j];

arr[i, j] = arr[iT, j];

arr[iT, j] = tempVal;

}

tempVal = avgArr[iT];

avgArr[iT] = avgArr[i];

avgArr[i] = tempVal;

}

}

}

return arr;

}

}

}

**Листинг класса VisualArray**

using System;

using System.Data;

namespace PW9\_02\_02\_ArrayDifSort

{

public class VisualArray

{

public DataTable CurrentTable { get; private set; } = new DataTable();

public DataTable CreateTable(double[,] arr)

{

CurrentTable = new DataTable();

for (int i = 0; i < arr.GetLength(1); i++)

{

CurrentTable.Columns.Add("Col\_" + (i + 1), typeof(string));

}

for (int i = 0; i < arr.GetLength(0); i++)

{

var row = CurrentTable.NewRow();

for (int j = 0; j < arr.GetLength(1); j++)

{

row[j] = arr[i, j];

}

CurrentTable.Rows.Add(row);

}

return CurrentTable;

}

public double[,] GetArray()

{

var arr = new double[CurrentTable.Rows.Count, CurrentTable.Columns.Count];

for (int i = 0; i < CurrentTable.Rows.Count; i++)

{

var row = CurrentTable.Rows[i];

for (int j = 0; j < CurrentTable.Columns.Count; j++)

{

arr[i, j] = Convert.ToDouble(row[j]);

}

}

return arr;

}

public void EditCell(int i, int j, double value)

{

var row = CurrentTable.Rows[i];//ПРОВЕРИТЬ АКТУАЛЬНОСТЬ РАБОТЫ!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

row[j] = value;

}

public void ClearTable()

{

CurrentTable = new DataTable();

}

}

}

**Классы, использующиеся в ArrayUnitTest**

**Листинг класса ArrayCreatorTest**

using CreatingArray;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace ArrayUnitTest

{

[TestClass]

public class ArrayCreatorTest

{

[TestMethod]

public void CreateArrayWith0()

{

double[,] arr0 = ArrayCreator.CreateArray(4, 4);

TraceTransfer.ToTrace(arr0);

Assert.AreEqual(0, arr0[0, 0]);

}

[TestMethod]

public void CreateArrayWithRandom()

{

double[,] arr0 = ArrayCreator.CreateArray(4, 4, new ArrayCreator.Range(2, 10));

TraceTransfer.ToTrace(arr0);

Assert.IsTrue(arr0[0, 0] != 0);

}

}

}

**Листинг класса ArraySorterTest**

using CreatingArray;

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace ArrayUnitTest

{

[TestClass]

public class ArraySorterTest

{

[TestMethod]//Смотреть трассировку отладки для этого метода

public void TestFirstWayElementsSort()

{

var sortArr = ArrayCreator.CreateArray(5, 5, new ArrayCreator.Range(1, 15));

TraceTransfer.ToTrace(sortArr);

sortArr = ArraySorter.FirstWaySortToUpRowElements(sortArr);

TraceTransfer.ToTrace(sortArr);

}

[TestMethod]//Смотреть трассировку отладки для этого метода

public void TestSecondWayElementsSort()

{

var sortArr = ArrayCreator.CreateArray(5, 5, new ArrayCreator.Range(1, 15));

TraceTransfer.ToTrace(sortArr);

sortArr = ArraySorter.SecondWaySortToUpRowElements(sortArr);

TraceTransfer.ToTrace(sortArr);

}

[TestMethod]//Смотреть трассировку отладки для этого метода

public void FirstWayRowSort()

{

var sortArr = ArrayCreator.CreateArray(5, 5, new ArrayCreator.Range(1, 15));

TraceTransfer.ToTrace(sortArr);

sortArr = ArraySorter.SecondWaySortToUpRowElements(sortArr);

TraceTransfer.ToTrace(sortArr);

var testAvg = ArraySorter.FindArrayAvgOfRows(sortArr);

TraceTransfer.ToTrace(testAvg);

sortArr = ArraySorter.FirstWaySortToUpRowsOnAvgElements(sortArr);

TraceTransfer.ToTrace(sortArr);

}

[TestMethod]//Смотреть трассировку отладки для этого метода

public void SecondWayRowSort()

{

var sortArr = ArrayCreator.CreateArray(5, 5, new ArrayCreator.Range(1, 15));

TraceTransfer.ToTrace(sortArr);

sortArr = ArraySorter.SecondWaySortToUpRowElements(sortArr);

TraceTransfer.ToTrace(sortArr);

var testAvg = ArraySorter.FindArrayAvgOfRows(sortArr);

TraceTransfer.ToTrace(testAvg);

sortArr = ArraySorter.SecondWaySortToUpRowsOnAvgElements(sortArr);

TraceTransfer.ToTrace(sortArr);

}

}

}

**Листинг TraceTransfer**

using System.Diagnostics;

namespace ArrayUnitTest

{

public static class TraceTransfer

{

public static void ToTrace(double[] avgArr)

{

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

{

Debug.Write($"{avgArr[i]} ");

}

Debug.WriteLine("");

Debug.WriteLine("");

}

public static void ToTrace(double[,] sortArr)

{

for (int i = 0; i < sortArr.GetLength(0); i++)

{

for (int j = 0; j < sortArr.GetLength(1); j++)

{

Debug.Write($"{sortArr[i, j]} ");

}

Debug.WriteLine("");

}

Debug.WriteLine("");

}

}

}