Тема 5 «Обработка одномерных массивов»

**Цель работы:** Овладеть основными приемами работы с таблицами для эффективной обработки массивов

**Задание 1. Разработка класса по обработке массива**

* Разработать программу обработки одномерного массива по заданию своего варианта. Методы по обработке массива поместите в отдельный класс. Выполните тестирование созданных методов с помощью тест-кейсов.

**Задание 2. Настройка пользовательского интерфейса**

* Ввод исходных данных должен осуществляться пользователем с клавиатуры или случайно. Размеры таблицы должны меняться интерактивно в определённых пределах.

**Задание 3. Обеспечение качества кода**

3.1 Самодокументируемый код

* Запустить проект
* Добавить XML-комментарии для всех функций
* Продемонстрировать всплывающую подсказку при вводе данной функции
* Выполнить выгрузку комментариев в XML-файл

3.2 Соглашение о кодировании

* Отредактировать код в соответствии с соглашением о кодировании

**Вариант 3**

* Заполнить одномерный массив целыми числами. Вычислить сумму и произведение элементов массива, индексы которых кратны заданному числу, а затем вставить после каждого такого элемента значение 1.

Выполнение задания

1. **Разработка подпрограмм**
   1. **Разработка программы Arr**
      1. **Математическая модель**

**Дано:** Массив целых чисел arr[], целое число numforcalcs, необходимое для поиска элементов массива, индексы которых кратны числу numforcalcs

**Найти:** modifiedarr, где modifiedarr[i + 1] = 1 и i % numforcalcs = 0, sum, mult Z

**Связь:** arr[i], mult = mult \* arr[i]

**ОДЗ:** дополнительных ограничений нет

* + 1. **Прототип**

/// <summary>

/// Метод заполняет массив случайными числами

/// </summary>

/// <param name="amount">кол.-во элементов массива</param>

/// <param name="random">Булево значение: случайное или ручное заполнение массива</param>

/// <returns>

/// Возращает массив, заполненный случайными числами или пустой массив

/// </returns>

public static int[] getArray(int amount, bool random)

/// <summary>

/// Метод вставляет новый элемент и присваивает ему значение "1" после элемента, индекс которого кратен заданному числу

/// </summary>

/// <param name="arr">исходный массив</param>

/// <param name="amount">кол.-во элементов в новом массиве</param>

/// <param name="numforcalcs">Число, для рассчета кратности индекса</param>

/// <returns>

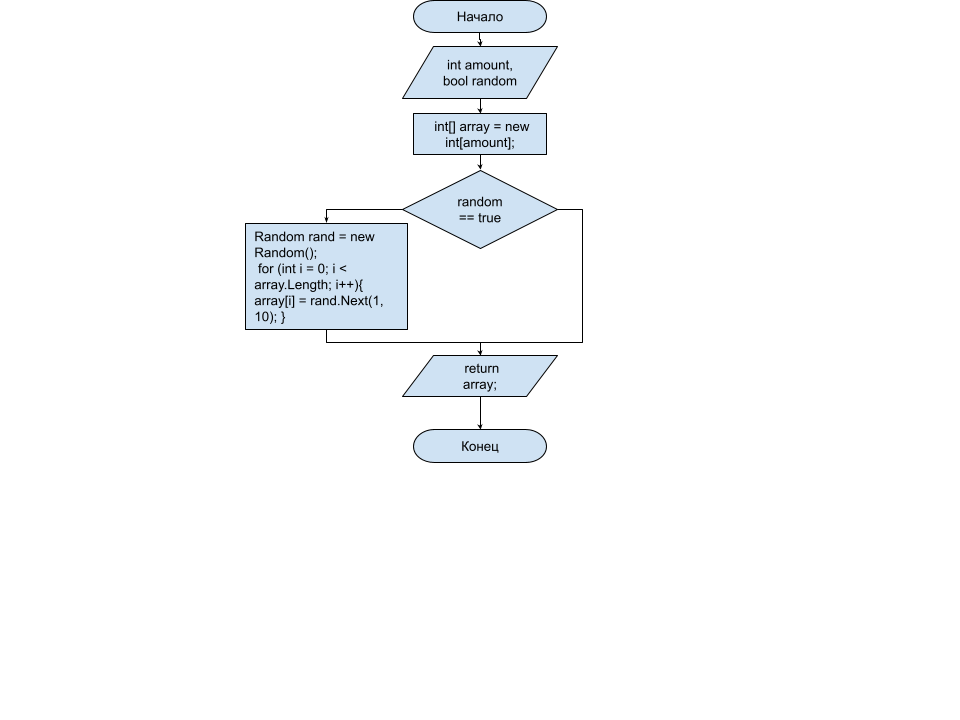
/// Возращает массив, заполненный случайными числами или пустой массив

/// </returns>

///

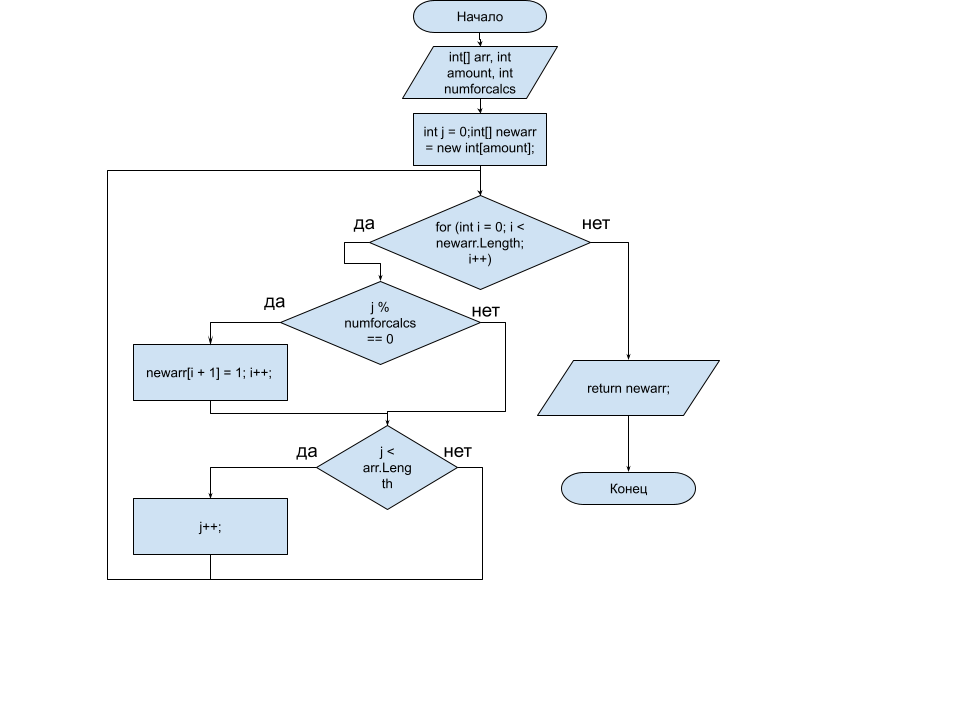
public static int[] getNewArray(int[] arr, int amount, int numforcalcs)

* + 1. **Схема работы подпрограммы**

****

Да Нет

**Рис.1.1.3.1 Метод getArray**

****

**Рис.1.1.3.2 Метод getNewArr**

* + 1. **Код программы** **Arr**

using System;

namespace prakt5\_Erdniev

{

public class Arr

{

/// <summary>

/// Метод заполняет массив случайными числами

/// </summary>

/// <param name="amount">кол.-во элементов массива</param>

/// <param name="random">Булево значение: случайное или ручное заполнение массива</param>

/// <returns>

/// Возращает массив, заполненный случайными числами или пустой массив

/// </returns>

public static int[] getArray(int amount, bool random)

{

int[] array = new int[amount];

if(random == true)

{

Random rand = new Random();

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

{

array[i] = rand.Next(1, 10);

}

}

return array;

}

/// <summary>

/// Метод вставляет новый элемент и присваивает ему значение "1" после элемента, индекс которого кратен заданному числу

/// </summary>

/// <param name="arr">исходный массив</param>

/// <param name="amount">кол.-во элементов в новом массиве</param>

/// <param name="numforcalcs">Число, для рассчета кратности индекса</param>

/// <returns>

/// Возращает массив, заполненный случайными числами или пустой массив

/// </returns>

///

public static int[] getNewArray(int[] arr, int amount, int numforcalcs)

{

int j = 0;

int[] newarr = new int[amount];

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

{

newarr[i] = arr[j];

if(j % numforcalcs == 0 )

{

newarr[i + 1] = 1;

i++;

}

if (j < arr.Length)

{

j++;

}

}

return newarr;

}

}

}

1. **Тестирование подпрограмм с помощью тест-кейсов**
   1. **Код модуля для тестов**

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using prakt5\_Erdniev;

namespace TestForPrakt5

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void Check\_getArray\_random()

{

int amount = 10;

bool random = true;

int[] arr = Arr.getArray(amount, random);

int[] emptyarr = new int[amount];

CollectionAssert.AreNotEqual(arr, emptyarr);

}

[TestMethod]

public void Check\_getArray\_manual()

{

int amount = 10;

bool random = false;

int[] arr = Arr.getArray(amount, random);

int[] emptyarr = new int[amount];

CollectionAssert.AreEqual(arr, emptyarr);

}

[TestMethod]

public void Check\_getNewArray1()

{

int amount = 10;

bool random = false;

int[] arr = Arr.getArray(amount, random);

int[] modifiedarr = Arr.getNewArray(arr, 15, 2);

int[] arrtocheck = new int[15] { 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0 };

CollectionAssert.AreEqual(modifiedarr, modifiedarr);

}

[TestMethod]

public void Check\_getNewArray2()

{

int amount = 10;

bool random = false;

int[] arr = Arr.getArray(amount, random);

int[] modifiedarr = Arr.getNewArray(arr, 14, 3);

int[] arrtocheck = new int[14] { 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1 };

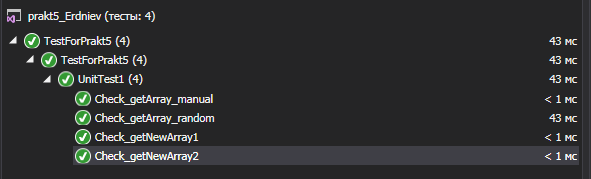
CollectionAssert.AreEqual(modifiedarr, modifiedarr);

}

}

}

* 1. **Результат выполнения тестов**



**Рис. 2.2.1 Результат тестов подпрограммы**

1. **Создание пользовательского интерфейса**
   1. **Проектирование пользовательского интерфейса**

Внешний вид проекта представлен на рис. 3.1.1, значения измененных свойств компонентов представлены в таблице 3.2.1

Изображение выглядит как текст

Автоматически созданное описание

groupBox1

Form1

randomly

manually

multOutput

sumOutput

Calculate

fillArrButton

Label2

Label1

ArrSize

numforcalcs

hint

dataGridView1

**Рис. 3.1.1**

* 1. **Свойства компонентов формы**

*Таблица 3.2.1* Значения свойств, установленные на этапе конструирования интерфейса

|  |  |  |
| --- | --- | --- |
| **Название компонента** | **Свойства** | **Значения** |
| **Form1** | **Text** | **Работа с одномерным массивом** |
| **FormBorderStyle** | **FixedSingle** |
| **Font** | **Microsoft Sans Serif; 8,25pt** |
| **groupBox1** | **Text** | **Способ заполнения массива** |
| **Randomly** | **Text** | **Случайно** |
| **Check** | **False** |
| **Manually** | **Text** | **Вручную** |
| **Check** | **False** |
| **Label1** | **Text** | **Укажите размер массива** |
| **Font** | **Microsoft Sans Serif; 8,25pt** |
| **Label2** | **Text** | **Укажите число** |
| **Font** | **Microsoft Sans Serif; 8,25pt** |
| **ArrSize** | **TabIndex** | **1** |
| **Max** | **100** |
| **Min** | **0** |
| **NumForCalcs** | **Font** | **Microsoft Sans Serif; 12pt** |
| **Enabled** | **False** |
| **Max** | **2** |
| **dataGridView1** | **ColumnHeadersHeight** | **23** |
| **hint** | **Text** | **Элементы массива с индексом, кратным числу будут использованы для вычислений** |
| **Visible** | **False** |
| **fillArrButton** | **Text** | **Заполнить массив** |
| **Enabled** | **False** |
| **Calculate** | **Text** | **Вычислить** |
| **Enabled** | **False** |
| **sumOutput** | **Text** | **Сумма:** |
| **Visible** | **False** |
| **multOutput** | **Text** | **Произведение:** |
| **Visible** | **False** |

* 1. **Сценарий использования программы**
     1. Указать размер массива.
     2. Заполнить массив случайными числами или вручную.
     3. Ввести число для расчетов
     4. Выполнение расчетов, изменение массива, вывод результатов
     5. Возврат к пунктам 1.1.1-1.1.3 по желанию пользователя
     6. Завершение работы программы.
  2. **Код модуля FormMain.cs**

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;

namespace prakt5\_Erdniev

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

fillArrButton.Enabled = false;

dataGridView1.RowCount = 1;

numForCalcs.Enabled = false;

}

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

{

int[] arr = new int[dataGridView1.ColumnCount];

int numforcalcs = int.Parse(numForCalcs.Text);

long sum = 0;

long mult = 1;

int appropriateindexes = 0;

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

arr[i] = int.Parse(dataGridView1.Rows[0].Cells[i].Value.ToString());

if (i % numforcalcs == 0)

{

sum += arr[i];

mult \*= arr[i];

appropriateindexes++;

}

appropriateindexes++;

}

sumOutput.Text = "Сумма: " + sum.ToString();

if (mult>long.MaxValue || mult <=0)

{

multOutput.Text = "Произведение: число слишком большое ";

}

else

{

multOutput.Text = "Произведение " + mult.ToString();

}

int[] modifiedarr = Array.getNewArray(arr, appropriateindexes, numforcalcs);

try

{

dataGridView1.ColumnCount = modifiedarr.Length;

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

{

dataGridView1.Columns[i].Name = i.ToString();

dataGridView1.Rows[0].Cells[i].Value = modifiedarr[i];

}

}

catch (Exception)

{

multOutput.Text = "Кол.-во элементов массива превышает доступный предел,\n заполните массив новыми числами";

calculate.Enabled = false;

}

}

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

{

int arrsize = int.Parse(arraySize.Value.ToString());

bool random = true;

if (randomly.Checked == true)

{

random = true;

} else if (randomly.Checked == false)

{

random = false;

}

int[] arr = Array.getArray(arrsize, random);

dataGridView1.ColumnCount = arrsize;

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

{

dataGridView1.Columns[i].Name = i.ToString();

dataGridView1.Rows[0].Cells[i].Value = arr[i];

}

numForCalcs.Enabled = true;

}

private void checkinput1()

{

if (randomly.Checked == true && arraySize.Value != 0) { fillArrButton.Enabled = true; } else if (manually.Checked == true && arraySize.Value != 0) { fillArrButton.Enabled = true; } else { fillArrButton.Enabled = false; }

}

private void arraySize\_ValueChanged(object sender, EventArgs e)

{

checkinput1();

}

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

{

checkinput1();

}

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

{

checkinput1();

}

private void numForCalcs\_MouseMove(object sender, MouseEventArgs e)

{

hint.Visible = true;

}

private void numForCalcs\_MouseLeave(object sender, EventArgs e)

{

hint.Visible = false;

}

private void numForCalcs\_TextChanged(object sender, EventArgs e)

{

if(numForCalcs.Text != "")

{

calculate.Enabled = true;

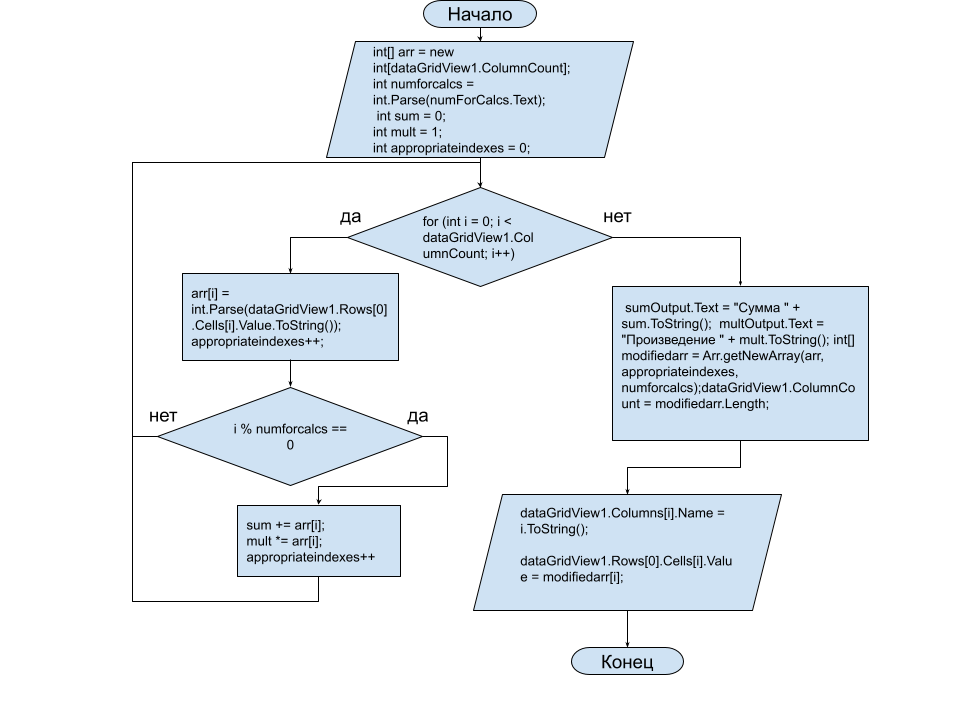
}

}

}

}

* 1. **Схемы основных подпрограмм**



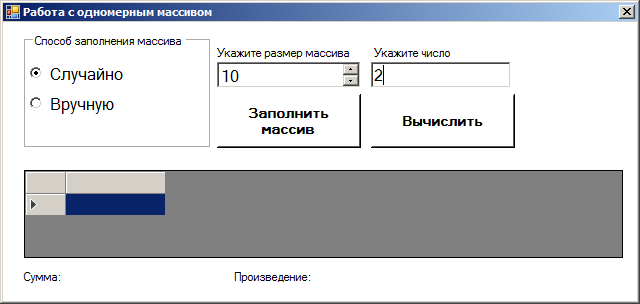
**Рис.3.5.1 Метод calculate\_Click**

* 1. **Протокол испытаний:**

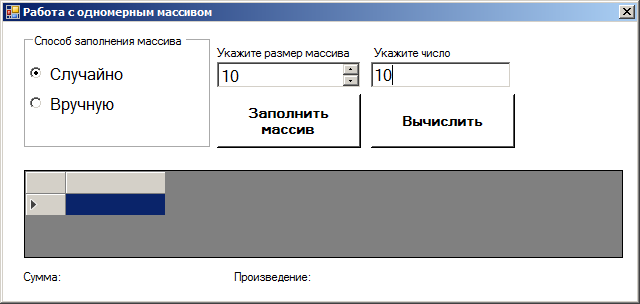
*Таблица 3.6.1.* Протокол испытаний

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Вариант входных данных** | **Ожидаемые результаты** | **Фактические результаты** |
| Способность контроля непереводимых полей ввода | Кол.-во элементов массива: 10  Заполнение массива: Случайные числа  Число для рассчетов:2 | Числа вводятся в поля | Соответствуют ожидаемым результатам  (Рис. 3.7.1) |
| Способность контроля вводимых данных | Кол.-во элементов массива: 10  Заполнение массива: Случайные числа  Число для рассчетов:1000 | В поле numForCalcs введено:10 | Соответствуют ожидаемым результатам  (Рис.3.7.2) |
| Способность контроля непереводимых полей ввода | Оставить поля ввода пустым | Кнопка для решения заблокирована | Соответствуют ожидаемым результатам  (Рис.3.7.3) |
| Проверить способность программы вычислять значения периметра многоугольника per | **Вариант 1**  Кол.-во элементов массива: 10  Заполнение массива: Случайные числа  Число для рассчетов:2  **Вариант 2**  Кол.-во элементов массива: 17  Заполнение массива: Случайные числа  Число для рассчетов:3 | Результат 1: Строка таблицы заполнена случайными числами  Выводятся рассчеты Sum, Mult  Результат 2: Строка таблицы заполнена случайными числами  Выводятся рассчеты Sum, Mult | Соответствуют ожидаемым результатам  (См. Рисунки 3.7.4-3.7.5) |

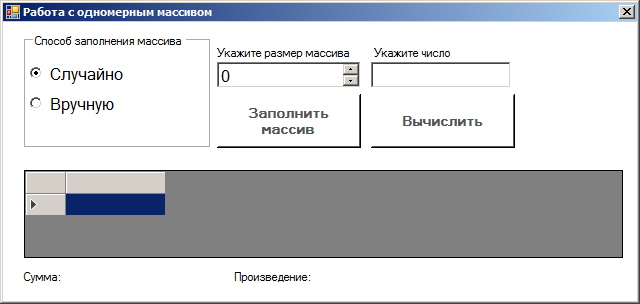
* 1. **Тестирование проекта:**



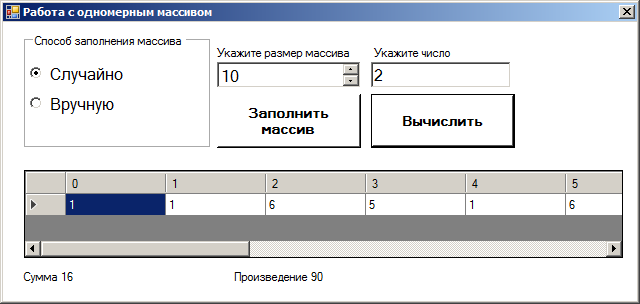
**Рис. 3.7.1**



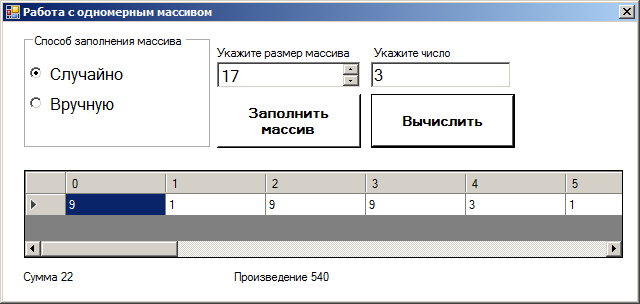
**Рис. 3.7.2**



**Рис. 3.7.3**

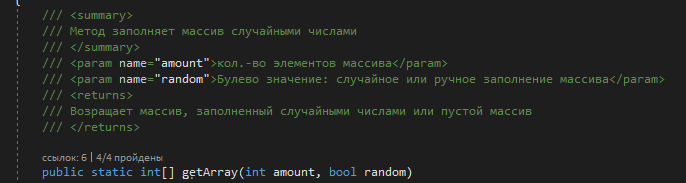


**Рис. 3.7.4**

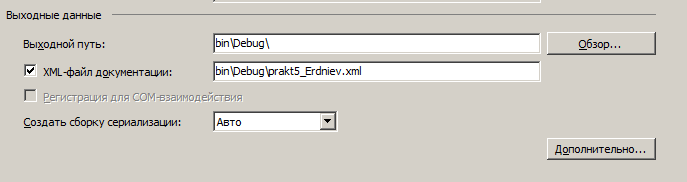


**Рис. 3.7.5**

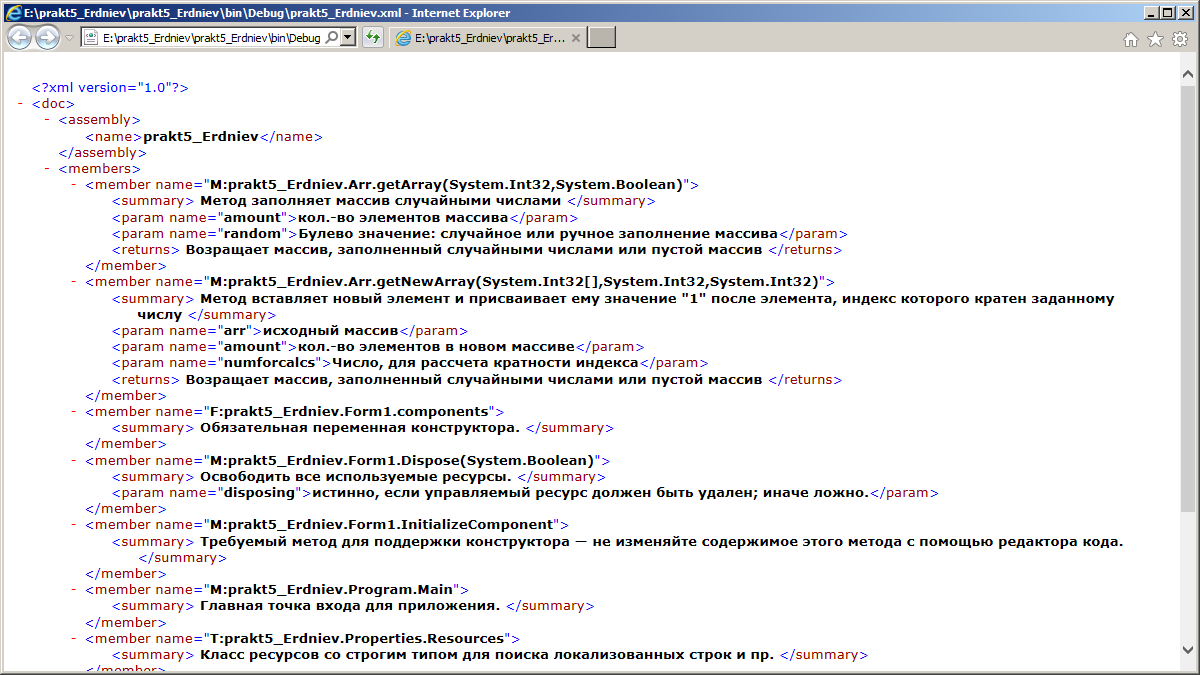
**4. Обеспечение качества кода**



**Рис. 4.1 XML комментарии к методам**



**Рис. 4.2 Путь к XML файлу**



**Рис. 4.3 XML файл**

**Код программы:**

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;

namespace prakt5\_Erdniev

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

fillArrButton.Enabled = false;

dataGridView1.RowCount = 1;

numForCalcs.Enabled = false;

}

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

{

int[] arr = new int[dataGridView1.ColumnCount];

int numforcalcs = int.Parse(numForCalcs.Text);

long sum = 0;

long mult = 1;

int appropriateindexes = 0;

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

arr[i] = int.Parse(dataGridView1.Rows[0].Cells[i].Value.ToString());

if (i % numforcalcs == 0)

{

sum += arr[i];

mult \*= arr[i];

appropriateindexes++;

}

appropriateindexes++;

}

sumOutput.Text = "Сумма: " + sum.ToString();

if (mult>long.MaxValue || mult <=0)

{

multOutput.Text = "Произведение: число слишком большое ";

}

else

{

multOutput.Text = "Произведение " + mult.ToString();

}

int[] modifiedarr = Array.getNewArray(arr, appropriateindexes, numforcalcs);

try

{

dataGridView1.ColumnCount = modifiedarr.Length;

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

{

dataGridView1.Columns[i].Name = i.ToString();

dataGridView1.Rows[0].Cells[i].Value = modifiedarr[i];

}

}

catch (Exception)

{

sumOutput.Text = "";

multOutput.Text = "Кол.-во элементов массива превышает доступный предел,\n заполните массив новыми числами";

calculate.Enabled = false;

}

}

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

{

int arrsize = int.Parse(arraySize.Value.ToString());

bool random = true;

if (randomly.Checked == true)

{

random = true;

} else if (randomly.Checked == false)

{

random = false;

}

int[] arr = Array.getArray(arrsize, random);

dataGridView1.ColumnCount = arrsize;

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

{

dataGridView1.Columns[i].Name = i.ToString();

dataGridView1.Rows[0].Cells[i].Value = arr[i];

}

numForCalcs.Enabled = true;

}

private void checkinput1()

{

if (randomly.Checked == true && arraySize.Value != 0) { fillArrButton.Enabled = true; } else if (manually.Checked == true && arraySize.Value != 0) { fillArrButton.Enabled = true; } else { fillArrButton.Enabled = false; }

}

private void arraySize\_ValueChanged(object sender, EventArgs e)

{

checkinput1();

}

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

{

checkinput1();

}

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

{

checkinput1();

}

private void numForCalcs\_MouseMove(object sender, MouseEventArgs e)

{

hint.Visible = true;

}

private void numForCalcs\_MouseLeave(object sender, EventArgs e)

{

hint.Visible = false;

}

private void numForCalcs\_TextChanged(object sender, EventArgs e)

{

if(numForCalcs.Text != "")

{

calculate.Enabled = true;

}

}

}

}

**Исправленный код программы по стандартам соглашения о кодировании:**

using System;

using System.Windows.Forms;

namespace prakt5\_Erdniev

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

fillArrButton.Enabled = false;

dataGridView1.RowCount = 1;

numForCalcs.Enabled = false;

}

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

{

int[] arr = new int[dataGridView1.ColumnCount];

int numforcalcs = int.Parse(numForCalcs.Text);

long sum = 0;

long mult = 1;

int appropriateindexes = 0;

for (int i = 0; i < dataGridView1.ColumnCount; i++)

{

arr[i] = int.Parse(dataGridView1.Rows[0].Cells[i].Value.ToString());

if (i % numforcalcs == 0)

{

sum += arr[i];

mult \*= arr[i];

appropriateindexes++;

}

appropriateindexes++;

}

sumOutput.Text = "Сумма: " + sum.ToString();

if (mult > long.MaxValue || mult <= 0)

{

multOutput.Text = "Произведение: число слишком большое ";

}

else

{

multOutput.Text = "Произведение " + mult.ToString();

}

int[] modifiedarr = Array.getNewArray(arr, appropriateindexes, numforcalcs);

try

{

dataGridView1.ColumnCount = modifiedarr.Length;

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

{

dataGridView1.Columns[i].Name = i.ToString();

dataGridView1.Rows[0].Cells[i].Value = modifiedarr[i];

}

}

catch (Exception)

{

sumOutput.Text = "";

multOutput.Text = "Кол.-во элементов массива превышает доступный предел,\n заполните массив новыми числами";

calculate.Enabled = false;

}

}

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

{

int arrsize = int.Parse(arraySize.Value.ToString());

bool random = true;

if (randomly.Checked == true)

{

random = true;

}

else if (randomly.Checked == false)

{

random = false;

}

int[] arr = Array.getArray(arrsize, random);

dataGridView1.ColumnCount = arrsize;

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

{

dataGridView1.Columns[i].Name = i.ToString();

dataGridView1.Rows[0].Cells[i].Value = arr[i];

}

numForCalcs.Enabled = true;

}

private void checkinput1()

{

if (randomly.Checked == true && arraySize.Value != 0) { fillArrButton.Enabled = true; } else if (manually.Checked == true && arraySize.Value != 0) { fillArrButton.Enabled = true; } else { fillArrButton.Enabled = false; }

}

private void arraySize\_ValueChanged(object sender, EventArgs e)

{

checkinput1();

}

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

{

checkinput1();

}

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

{

checkinput1();

}

private void numForCalcs\_MouseMove(object sender, MouseEventArgs e)

{

hint.Visible = true;

}

private void numForCalcs\_MouseLeave(object sender, EventArgs e)

{

hint.Visible = false;

}

private void numForCalcs\_TextChanged(object sender, EventArgs e)

{

if (numForCalcs.Text != "")

{

calculate.Enabled = true;

}

}

}

}