**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.

**Вариант 5:** заполнить одномерный массив целыми числами. Вычислить номер первого кратного 3 и последнего кратного 3 элемента массива и поменять их в массиве местами. Вычислить также сумму элементов, стоящих между ними. Если кратных трём нет или только один, то вывести сообщение об этом.

**Код класса IntArray:**

public class IntArray

{

public int[] array;

private int length;

public int firstIndex = -1;

public int lastIndex = -1;

/// <summary>

/// Конструктор для создания экземпляра пустого массива

/// </summary>

/// <param name="length">присвоение длины массива экземпляра классу</param>

public IntArray(int length)

{

if (length < 0)

{

throw new ArgumentOutOfRangeException(nameof(length), "Указано неверное значение длины массива.");

}

this.length = length;

array = new int[length];

}

/// <summary>

/// Конструктор для получения массива в класс

/// </summary>

/// <param name="arr">присвоение массива экземпляра классу</param>

public IntArray(params int[] arr)

{

length = arr.Length;

array = arr;

}

/// <summary>

/// Кнопка "Вычислить"- запускает метод вычисления среднего значения положительных элементов.

/// </summary

public int Length

{

get

{

return length;

}

}

/// <summary>

/// Метод для заполнения массива рандомными значения

/// </summary>

/// <param name="a">первое число диапазона для рандомных чисел</param>

/// <param name="b">второе число диапазона для рандомных чисел</param>

public void RandomIntArray( int a, int b)

{

Random rand = new Random();

for (int i = 0; i < length; i++)

{

array[i] = rand.Next(a, b + 1);

}

}

/// <summary>

/// Вычисление номера первого кратного 3 и последнего кратного 3 элемента массива и изменение их в массиве местами

/// </summary>

public void first\_last\_ind()

{

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

{

if (array[i] % 3 == 0)

{

if (firstIndex == -1)

{

firstIndex = i;

}

lastIndex = i;

}

}

if (firstIndex == -1 || lastIndex == -1)

{

throw new Exception("Кратные трём элементы отсутствуют.");

}

int temp = array[firstIndex];

array[firstIndex] = array[lastIndex];

array[lastIndex] = temp;

}

/// <summary>

/// Вычисление произведения элементов, стоящих между первым кратным 3 и последнего кратным 3 элементами массива.

/// </summary>

public int SumArray()

{

int pr = 0;

for (int i = firstIndex + 1; i < lastIndex; i++)

{

pr \*= array[i];

}

return pr;

}

}

**Unit Test для проверки методов класса IntArray:**

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethoProisved()

{

int[] A = {-9, 5, 0, 3, -4, 8, 2};

IntArray arr = new IntArray(A);

double expected = 0;

arr.first\_last\_ind();

int pr = arr.SumArray();

Assert.AreEqual(expected,pr,"Ожидаемое значение не получено.");

}

TestMethod]

public void TestMethodPerestan()

{

try

{

int[] A = { 1, 5, 4, 1, -4, 8, 2 };

int[] expected = { 3, 5, 4,-9, -4, 8, 2 };

IntArray arr = new IntArray(A);

arr.first\_last\_ind();

int[] B = arr.array;

// Assert.AreEqual(expected, B, "Ожидаемое значение не получено.");

}

catch (Exception ex)

{

StringAssert.Contains(ex.Message, "Кратные трём элементы отсутствуют.");

return;

}

Assert.Fail("Ожидаемое исключение не получено.");

}

[TestMethod]

public void TestMethodfirstindex()

{

int[] A = { 1, 5, 3, 1, -4, 6, 2 };

IntArray arr = new IntArray(A);

int expected =2;

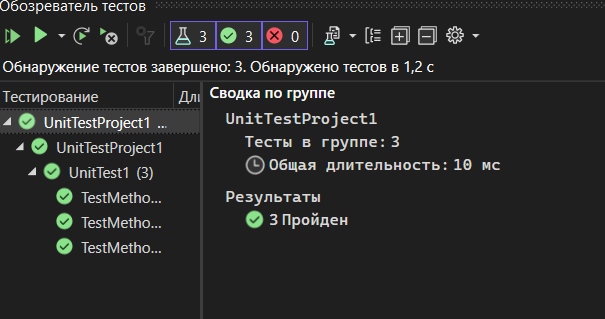
arr.first\_last\_ind();

int B = arr.firstIndex;

Assert.AreEqual(expected, B, "Ожидаемое значение не получено.");

}

}

****

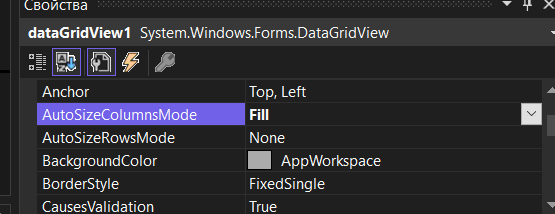
**Рис. 1.1 – Результат запуска тестов**

**Задание 2.** Настройка пользовательского интерфейса

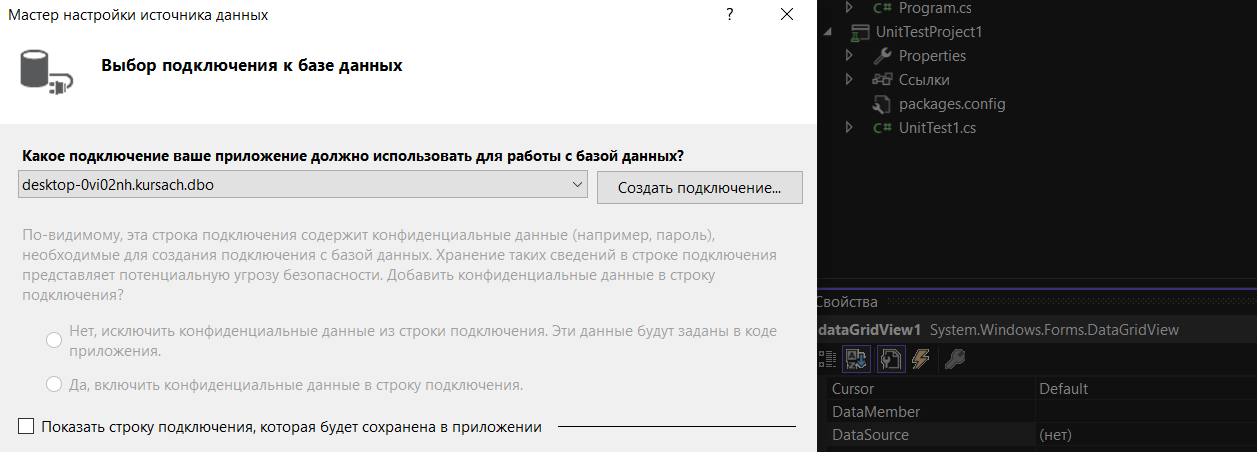
Ввод исходных данных должен осуществляться пользователем с клавиатуры или случайно. Размеры таблицы должны меняться интерактивно в определённых пределах.

Основные настройки пользовательского интерфейса:

Таблица представляет собой специальный компонент, позволяющие отображать данные в виде строк и столбцов. Компонент dataGridView позволяет хранить и отображать текстовую и графическую информацию. Но хранение и отображение данных выполняется программистом. Есть возможность подключить источник данных, которые будут представляться в таблице.



**Рис. 2.1 – настройка интерактивного размера**

****

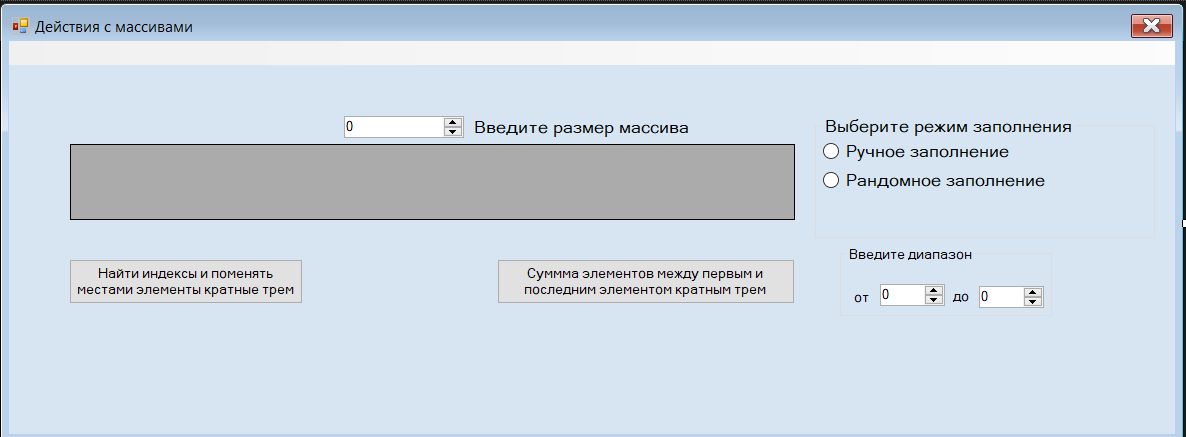
**Рис. 2.2 – возможность подключения к базе данных**

Добавьте в проект проверку нажатых клавиш в процессе редактирования каждой ячейки таблицы. Запретить ввод недопустимых символов при редактировании в ручном режиме.

Если пользователь вручную вводит данные в таблицу, то велика вероятность ошибок при вводе. Поэтому желательно заблокировать недопустимые символы и добавить проверку вводимых в каждую ячейку данных, что сведёт количество ошибок ввода к минимуму. Для этого воспользуемся событием KeyPress для таблицы. Событие проверки нажимаемых клавиш должно вызываться при редактировании каждой ячейки таблицы, из события EditingControlShowing.

Внешний вид проекта представлен на рис. 2.3, значения измененных свойств компонентов представлены в таблице 2.1.

Form1



GroupBox2

Radiobutton2

radiobutton1

GroupBox1

datagridview1

Numericupdown3

Numericupdown2

Numericupdown11

Label2

button1

Button2

**Рис. 2.3 – внешний вид формы**

Label1

button1

* 1. **Свойства компонентов формы**

*Таблица 2.1.* Значения свойств, установленные на этапе конструирования интерфейса

|  |  |  |
| --- | --- | --- |
| **Название компонента** | **Свойства** | **Значения** |
| form1 | Text | Действия с массивами |
| BackColor | GradientInactiveCaption |
| Font | 9 |
| Начертание: | обычный |
| button1 | Text | «Найти индексы и поменять местами элементы кратные трем» |
| button2 | Text | «Сумма элементов между первым и последним элементом кратным трем» |
| label2 | Text | «Введите размер массива» |
| label1 | Font | 9 |
| datagridview1 | AutoSizeColumsMode | Fill |
| groupBox1 | Text | «Выберите режим заполнения» |
| groupBox2 | Text | «Введите диапазон» |
| Radiobutton1 | Text | «Ручное заполнение» |
| RadioButton2 | Text | «Рандомное заполнение» |

* 1. **Сценарий использования программы**
     1. Определение размера массива
     2. Выбор режима заполнения данных
        1. Если выбор «Ручное заполнение»
        2. Двойное нажатие на datagridview1
        3. Заполнение данных вручную
        4. Если выбор «Ручное заполнение»
        5. Определить диапазон рандомных значений
        6. Двойное нажатие на datagridview1
     3. Нажать на button 1
     4. Ознакомиться с результатом
     5. Нажать на button 2
     6. Ознакомиться с результатом
     7. Возможность вернуться к предыдущим пунктам
     8. Завершение работы программы

**Код класса Form1:**

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WindowsFormsApp1

namespace lab21

{

public partial class Form1 : Form

{

private IntArray arr1;

private IntArray arr2;

private IntArray arr\_res;

Random r = new Random();

int ot;

int doc;

/// <summary>

/// Конструктор формы

/// </summary>

public Form1()

{

InitializeComponent();

}

/// <summary>

/// Преобразование в массив данных с datagridview создание экземпляров класса с методами

/// </summary>

private void ConvertToArray1()

{

int col = Convert.ToInt32(numericUpDown1.Value);

int [] arr = new int [col];

for (int i = 0; i < col; i++)

{

if (dataGridView1.Rows[0].Cells[i].Value != null)

{

arr[i] = Convert.ToInt32(dataGridView1.Rows[0].Cells[i].Value);

}

else

{

MessageBox.Show("Ошибка ввода данных в столбце" + i);

return;

}

}

arr1 = new IntArray(arr);

}

/// <summary>

/// Метод двойного нажатия на datagridview, который создает ячейки и столбцы, загружает данные и т.д., проверяет значения

/// </summary>

/// <param name="sender">указатель на button1</param>

/// <param name="e">дополнительный аргумент</param>

private void dataGridView1\_DoubleClick(object sender, EventArgs e)

{

int col = Convert.ToInt32(numericUpDown1.Value);

if (col == 0)

{

dataGridView1.ColumnCount = 0; ;

dataGridView1.RowCount = 0;

MessageBox.Show("Неверный формат данных.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

dataGridView1.ColumnCount = col;

dataGridView1.RowCount = 1;

try

{

if (radioButton1.Checked)

{

dataGridView1.RowCount = 1;

for (int i = 0; i < col; i++)

{

dataGridView1.Columns[i].Name = (i + 1).ToString();

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

}

dataGridView1.ReadOnly = false;

}

else if (radioButton2.Checked)

{

ot = Convert.ToInt32(numericUpDown3.Value);

doc = Convert.ToInt32(numericUpDown4.Value);

dataGridView1.ReadOnly = true;

arr1 = new IntArray(col);

int[] arr;

arr1.RandomIntArray(ot, doc);

arr= arr1.array;

for (int i = 0; i < col; i++)

{

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

dataGridView1.Rows[0].Cells[i].Value = Convert.ToInt32(arr[i]);

}

}

else

{

dataGridView1.ReadOnly = true;

}

}

catch (System.FormatException)

{

MessageBox.Show("Неверный формат данных.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Метод выбора кнопки "Рандомное заполнение", очищает datagridview

/// </summary>

/// <param name="sender">указатель на radioButton2</param>

/// <param name="e">дополнительный аргумент</param>

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

dataGridView1.Columns.Clear();

dataGridView1.Rows.Clear();

}

/// <summary>

/// Метод выбора кнопки "Ручное заполнение", очищает datagridview

/// </summary>

/// <param name="sender">указатель на radioButton1</param>

/// <param name="e">дополнительный аргумент</param>

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

dataGridView1.Columns.Clear();

dataGridView1.Rows.Clear();

}

/// <summary>

/// Метод проверки в datagridview

/// </summary>

/// <param name="sender">указатель на dataGridView1</param>

/// <param name="e">дополнительный аргумент</param>

private void dataGridView1\_CellValidating(object sender, DataGridViewCellValidatingEventArgs e)

{

if (e.RowIndex >= 0 && e.ColumnIndex >= 0)

{

int value;

if (!int.TryParse(e.FormattedValue.ToString(), out value))

{

MessageBox.Show("Введите корректное числовое значение.", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

e.Cancel = true;

}

}

}

/// <summary>

/// Метод проверки в datagridview

/// </summary>

/// <param name="sender">указатель на dataGridView1</param>

/// <param name="e">дополнительный аргумент</param>

private void dataGridView1\_EditingControlShowing(object sender, DataGridViewEditingControlShowingEventArgs e)

{

e.Control.KeyPress += new KeyPressEventHandler(dataGridView1\_KeyPress);

}

/// <summary>

/// Метод контроля нажатия клавиш в datagridview

/// </summary>

/// <param name="sender">указатель на dataGridView1</param>

/// <param name="e">дополнительный аргумент</param>

private void dataGridView1\_KeyPress(object sender, KeyPressEventArgs e)

{

bool hasMin = Text.Contains('-');

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) &&(e.KeyChar != '-' ))

{

e.Handled = true;

}

if ( (e.KeyChar == '-' && hasMin))

{

e.Handled = true;

}

}

/// <summary>

/// Метод нажатия на кнопку "Найти индексы и поменять местами элементы кратные трем"

/// </summary>

/// <param name="sender">указатель на button1</param>

/// <param name="e">дополнительный аргумент</param>

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

{

try

{

ConvertToArray1();

arr1.first\_last\_ind();

int g =arr1.firstIndex;

int gs = arr1.lastIndex;

for (int i = 0; i < arr1.Length; i++)

{

dataGridView1.Columns[i].Name = (i + 1).ToString();

dataGridView1.Rows[0].Cells[i].Value = Convert.ToInt32(arr1.array[i]);

}

label1.Text = string.Format("Первый индекс элемента кратного трём {0} \r\nПоследний индекс элемента кратного трём {1}", arr1.firstIndex, arr1.lastIndex);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Метод нажатия на кнопку "Суммма элементов между первым и последним элементом кратным трем"

/// </summary>

/// <param name="sender">указатель на button2</param>

/// <param name="e">дополнительный аргумент</param>

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

{

try

{

ConvertToArray1();

arr1.first\_last\_ind();

int a = arr1.SumArray();

label1.Text = string.Format("Первый индекс элемента кратного трём {0} \r\nПоследний индекс элемента кратного трём {1}\r\nСумма элементов стоящих между ними {2}", arr1.firstIndex, arr1.lastIndex, a);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

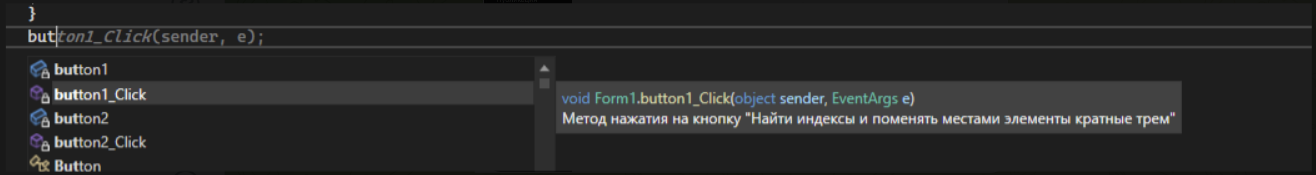
}

**Задание 3.** Обеспечение качества кода

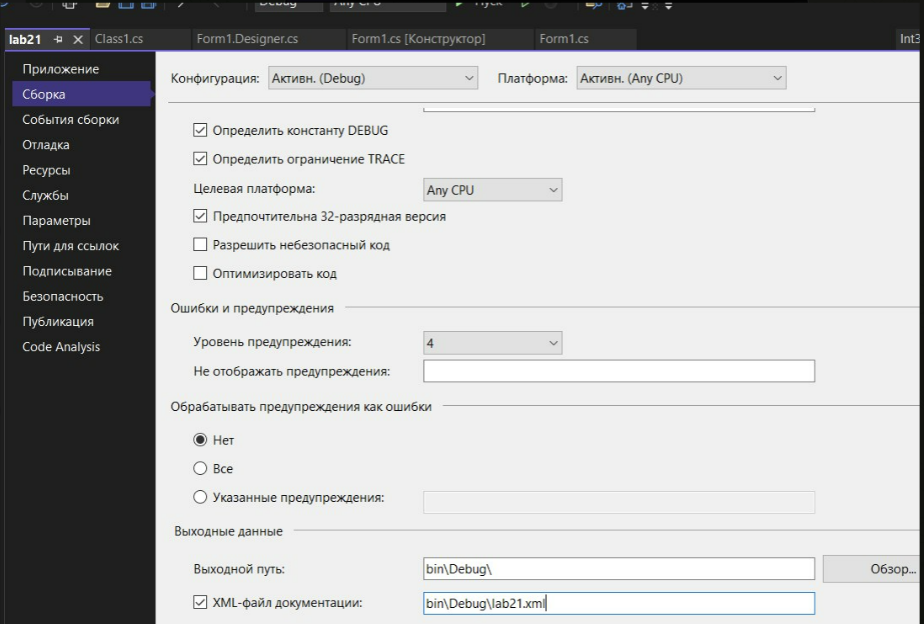
**3.1** Самодокументируемый код

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

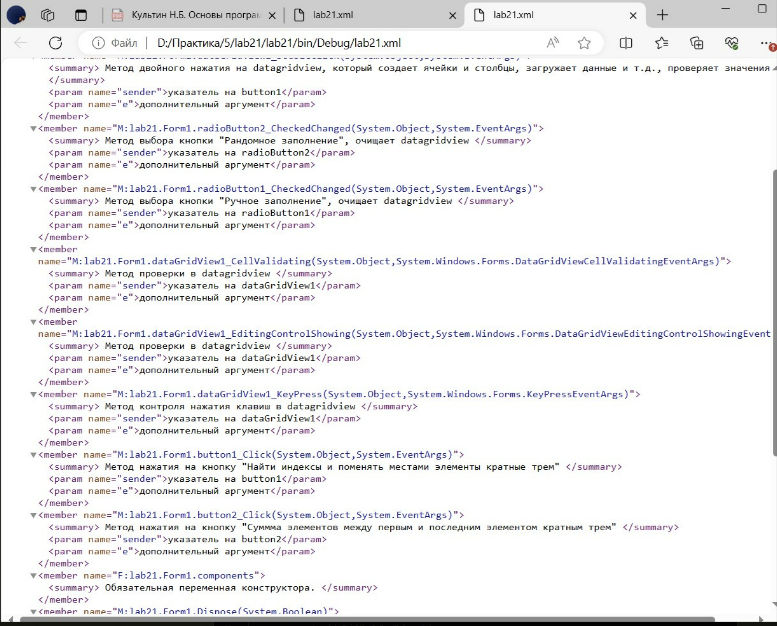
XML-комментарии см код Form1, class IntArray. Результат представлен на рисунках .

****

**Рис. 3.1.1 – Всплывающая подсказка**

****

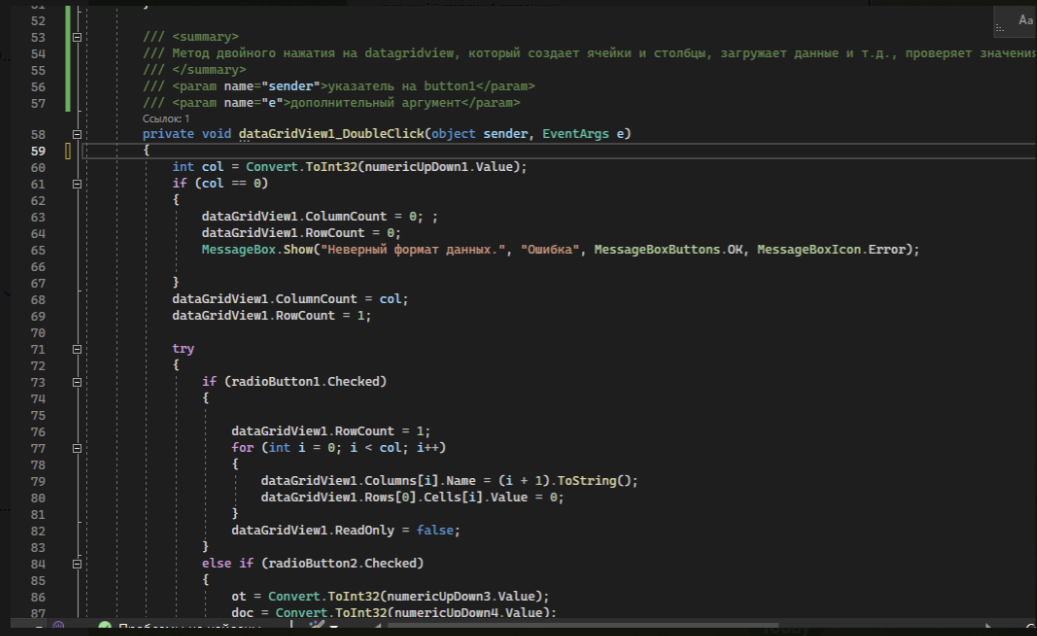
**Рис. 3.1.2 – Выгрузка комментариев в XML-файл**

****

**Рис. 3.1.3 – файл с XML-комментариями**

**3.2** Соглашение о кодировании

* Отредактировать код в соответствии с соглашением о кодировании

****

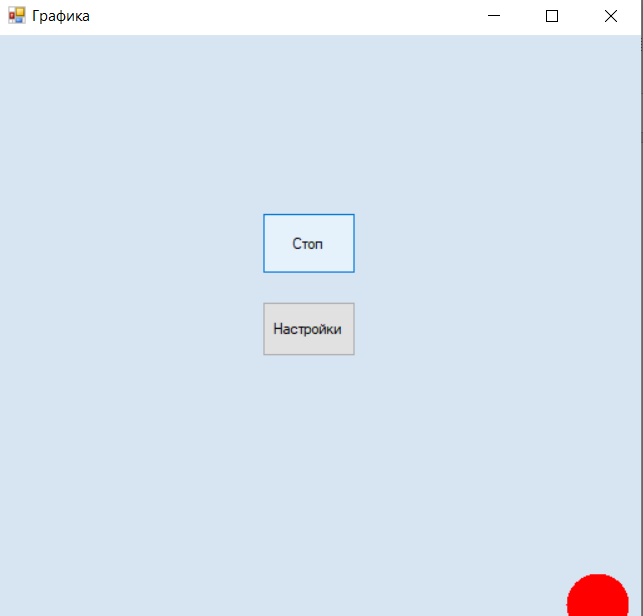
**Рис. 3.2.1 – Отредактированный код**

**3. Протокол испытаний:**

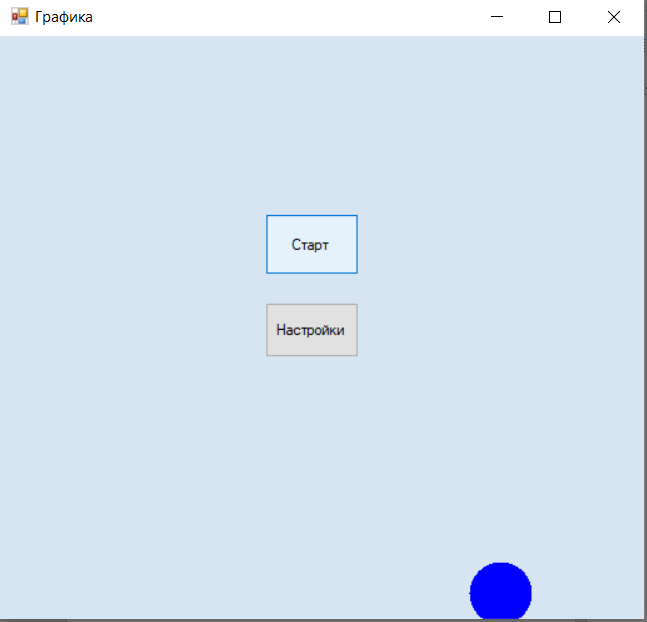
*Таблица 3.1.* Протокол испытаний

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Вариант входных данных** | **Ожидаемые результаты** | **Фактические результаты** |
| Изменение названия кнопки | Фигура не движется, и фигура движется | Когда фигура в движении кнопка имеет названия «Стоп», при остановке имеет название «Старт» | Соответствуют ожидаемым результатам  (См. Рисунки 3.1.1-3.1.2) |
| Изменение скорости движения | Число вводится в numericupdown от 1 до 10000 | Чем больше число, тем медленнее движение эллипса | Соответствуют ожидаемым результатам  (См. Рисунок 3.1.3) |
| Установка нового цвета фигуры для прямого движения | Выбор цвета в colordialog | Фигура окрашивается в выбранный цвет и поддерживает его при прямом движении | Соответствуют ожидаемым результатам  (См. Рисунки 3.1.4 – 3.1.5) |
| Установка нового цвета фигуры для обратного движения | Выбор цвета в colordialog | Фигура окрашивается в выбранный цвет при обратном движении | Соответствуют ожидаемым результатам  (См. Рисунки 3.1.6 – 3.1.7) |
| Способность фигуры двигаться по периметру формы, не учитывая верхней границы | Наблюдение за работой | Фигура движется по трем сторонам | Соответствует ожидаемым результатам  (См. Рисунок 3.1.8) |
| Способность закрытия формы по клавише Esc | Нажать на кнопку Esc при работе формы | Приложение прекращает работу | Соответствует ожидаемым результатам |

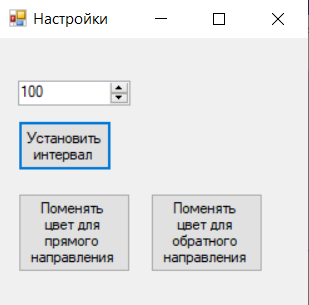
* 1. **Результаты тестирования проекта:**

****

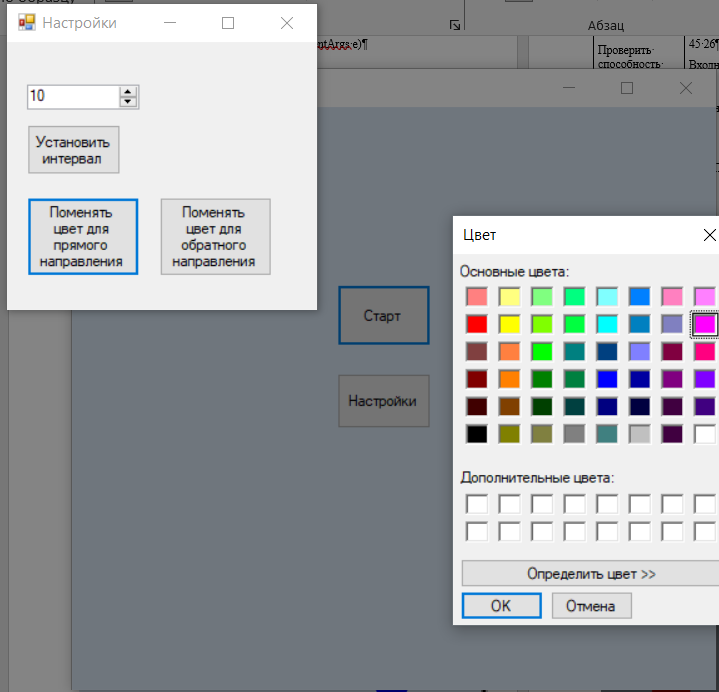
**Рис. 3.1.1**

****

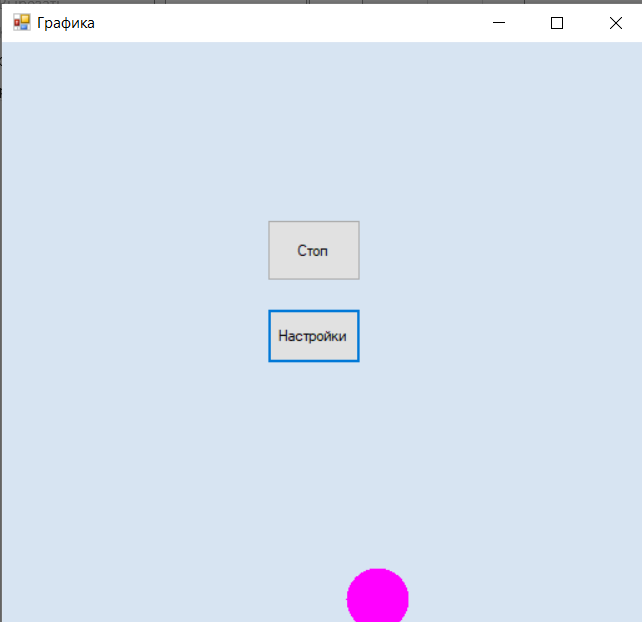
**Рис. 3.1.2**

****

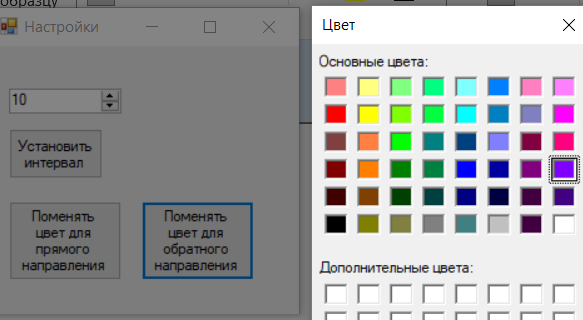
**Рис. 3.1.3**

****

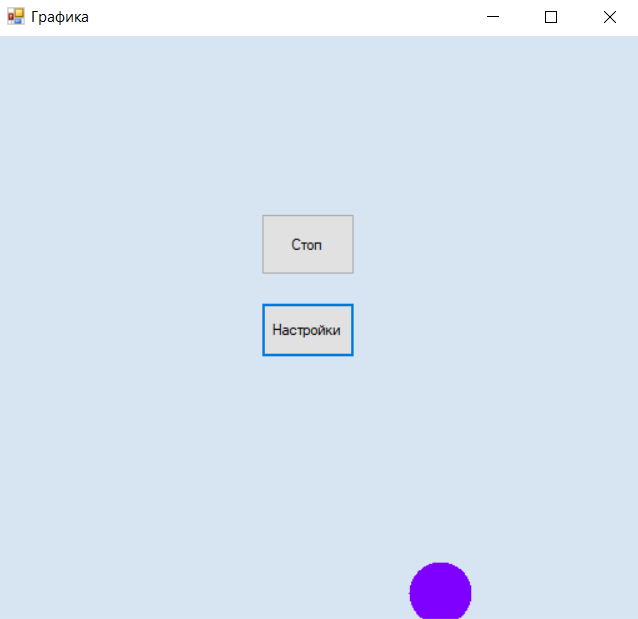
**Рис. 3.1.4**

****

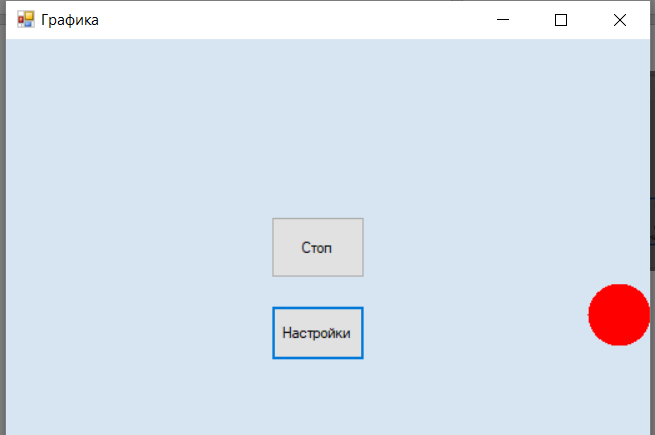
**Рис. 3.1.5**

****

**Рис. 3.1.6**

****

**Рис. 3.1.7**

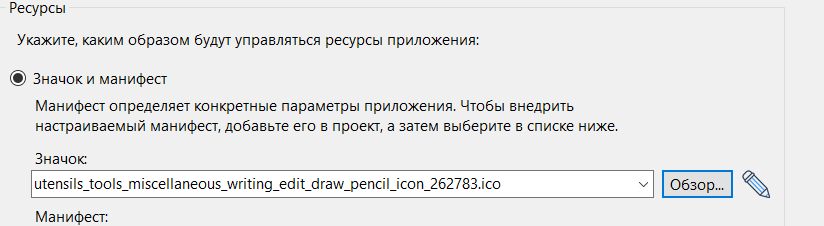
****

**Рис. 3.1.8**

**Задание 3.** Инсталляция приложения

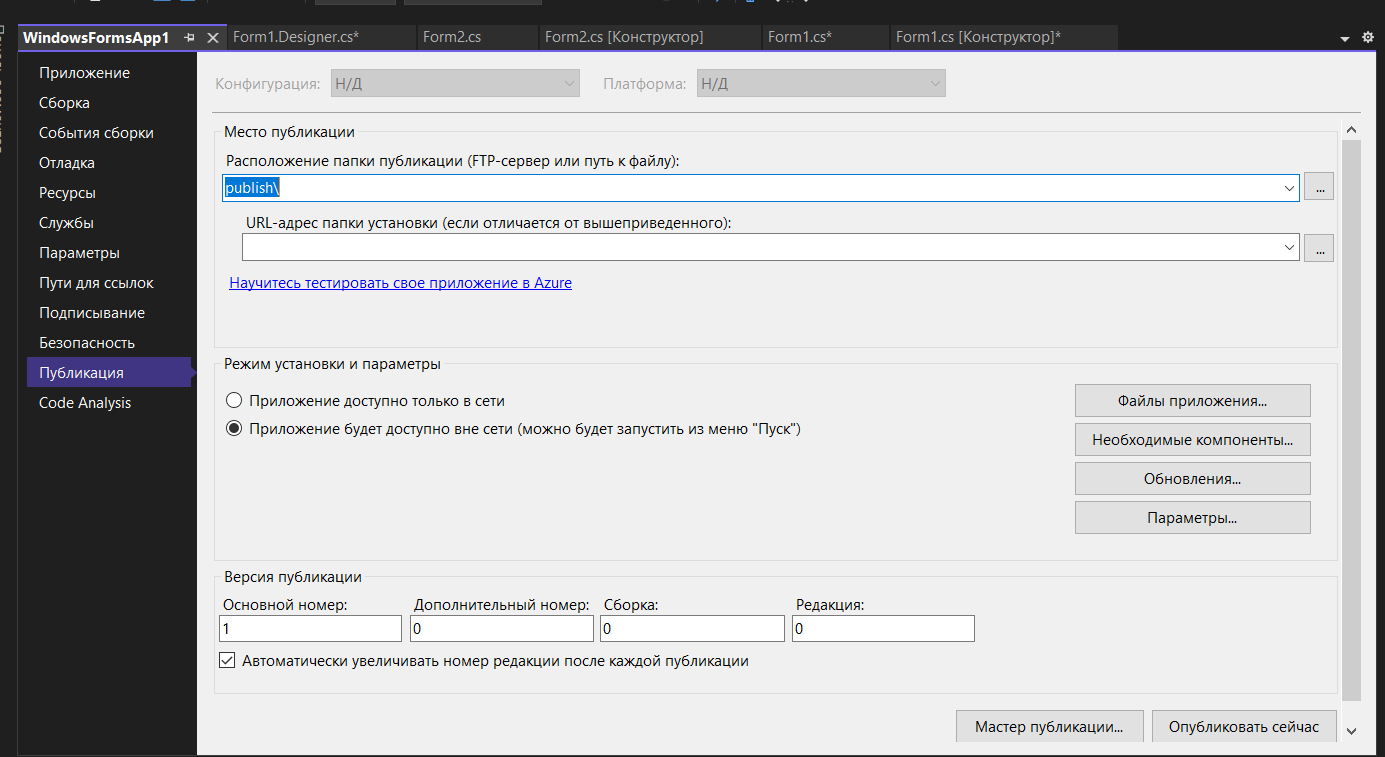
Добавить иконку в проект и название проекта. Для созданного вами приложения провести инсталляцию на свой компьютер. Инсталляция происходит при помощи мастера публикации (находится в свойствах проекта).

Установка иконки приложения (см. Рисунок 3.2.1)

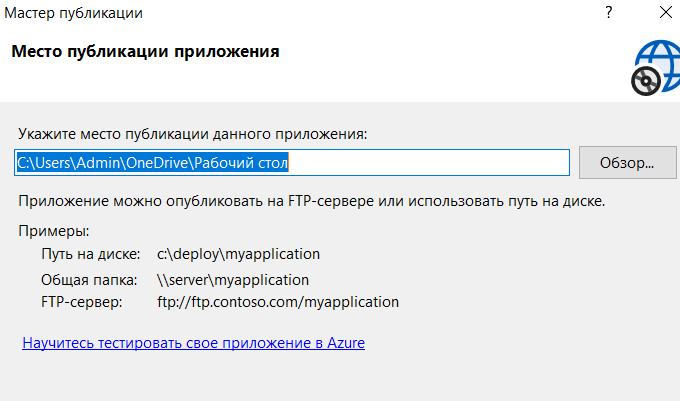
****

**Рис. 3.2.1**

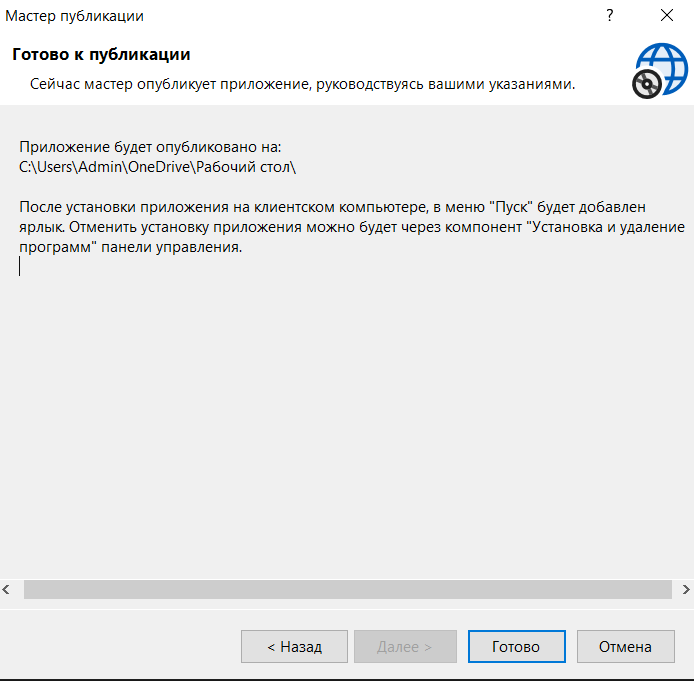
Работа с мастером публикаций (см. Рисунки 3.2.2 – 3.2.6)

****

**Рис 3.2.2 – начало работы мастера публикаций**



**Рис 3.2.3 – выбор места установки**



**Рис 3.2.4 – готовность инсталлятора**

