**23 Элементы управления.**

Задание 1. Разработать программу, добавляющую на форму метки с текстом. Местоположение и размеры меток определяются в программе динамически через поля ввода. В заголовок окна, анализируя размер всех меток, вывести количество маленьких и больших меток. Маленькой меткой считается метка размером менее 50 пикселей по горизонтали и по вертикали.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace WinFormsApp1

{

public partial class Form1 : Form

{

private TextBox textBoxText;

private TextBox textBoxX;

private TextBox textBoxY;

private TextBox textBoxWidth;

private TextBox textBoxHeight;

private List<Label> labels;

public Form1()

{

InitializeComponent();

labels = new List<Label>();

}

private void Form1\_Load(object sender, EventArgs e)

{

AddControls();

}

private void AddControls()

{

Label labelText = new Label

{

Text = "Текст метки:",

Location = new System.Drawing.Point(20, 20),

Size = new System.Drawing.Size(100, 20)

};

textBoxText = new TextBox

{

Location = new System.Drawing.Point(130, 20),

Size = new System.Drawing.Size(100, 20)

};

Label labelX = new Label

{

Text = "X:",

Location = new System.Drawing.Point(20, 50),

Size = new System.Drawing.Size(20, 20)

};

textBoxX = new TextBox

{

Location = new System.Drawing.Point(50, 50),

Size = new System.Drawing.Size(50, 20)

};

Label labelY = new Label

{

Text = "Y:",

Location = new System.Drawing.Point(110, 50),

Size = new System.Drawing.Size(20, 20)

};

textBoxY = new TextBox

{

Location = new System.Drawing.Point(140, 50),

Size = new System.Drawing.Size(50, 20)

};

Label labelWidth = new Label

{

Text = "Ширина:",

Location = new System.Drawing.Point(200, 50),

Size = new System.Drawing.Size(80, 20)

};

textBoxWidth = new TextBox

{

Location = new System.Drawing.Point(300, 50),

Size = new System.Drawing.Size(50, 20)

};

Label labelHeight = new Label

{

Text = "Высота:",

Location = new System.Drawing.Point(400, 50),

Size = new System.Drawing.Size(80, 20)

};

textBoxHeight = new TextBox

{

Location = new System.Drawing.Point(500, 50),

Size = new System.Drawing.Size(50, 20)

};

Button button = new Button

{

Text = "Добавить",

Location = new System.Drawing.Point(20, 80),

Size = new System.Drawing.Size(100, 30)

};

button.Click += ButtonAddLabel\_Click;

Controls.Add(labelText);

Controls.Add(textBoxText);

Controls.Add(labelX);

Controls.Add(textBoxX);

Controls.Add(labelY);

Controls.Add(textBoxY);

Controls.Add(labelWidth);

Controls.Add(textBoxWidth);

Controls.Add(labelHeight);

Controls.Add(textBoxHeight);

Controls.Add(button);

}

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

{

// Получаем значения из текстовых полей

string labelText = textBoxText.Text;

int x = int.Parse(textBoxX.Text);

int y = int.Parse(textBoxY.Text);

int width = int.Parse(textBoxWidth.Text);

int height = int.Parse(textBoxHeight.Text);

// Создаем новую метку

Label newLabel = new Label

{

Text = labelText,

Location = new System.Drawing.Point(x, y),

Size = new System.Drawing.Size(width, height),

BorderStyle = BorderStyle.FixedSingle

};

labels.Add(newLabel);

Controls.Add(newLabel);

// Обновляем заголовок формы

UpdateTitle();

}

private void UpdateTitle()

{

int smallLabelsCount = labels.Count(l => l.Width < 50 && l.Height < 50);

int largeLabelsCount = labels.Count(l => l.Width >= 50 || l.Height >= 50);

this.Text = $"Маленькие метки: {smallLabelsCount}, Большие метки: {largeLabelsCount}";

}

}

}

Таблица 23.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Метка, 200, 200, 200, 200 | Маленькие метки:0, Большие метки:1 |

Анализ результатов:

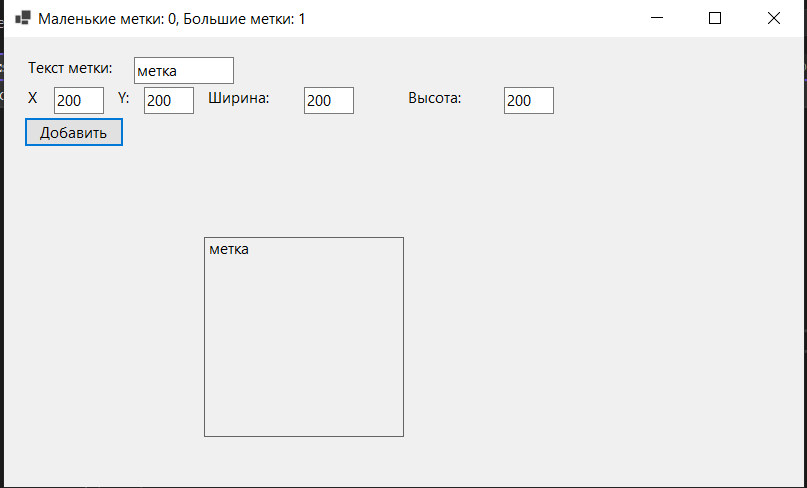


Рисунок 23.1 – Результат работы программы

Задание 2. Построить график функции. Таблицу данных получить путём изменения параметра X с шагом h. Самостоятельно выбрать удобные параметры настройки. y=e^x

Листинг кода:

using System;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace WinFormsApp2

{

public partial class Form1 : Form

{

private Chart chart;

private TextBox textBoxStep;

private Button buttonPlot;

public Form1()

{

InitializeComponent();

InitializeChart();

}

private void InitializeChart()

{

// Создаем и настраиваем элемент управления Chart

chart = new Chart

{

Location = new System.Drawing.Point(20, 60),

Size = new System.Drawing.Size(600, 400)

};

ChartArea chartArea = new ChartArea

{

Name = "ChartArea1"

};

chart.ChartAreas.Add(chartArea);

Series series = new Series

{

Name = "Series1",

ChartType = SeriesChartType.Line,

ChartArea = "ChartArea1"

};

chart.Series.Add(series);

Controls.Add(chart);

// Добавляем текстовое поле для ввода шага

Label labelStep = new Label

{

Text = "Шаг h:",

Location = new System.Drawing.Point(20, 20),

Size = new System.Drawing.Size(50, 20)

};

textBoxStep = new TextBox

{

Location = new System.Drawing.Point(80, 20),

Size = new System.Drawing.Size(100, 20)

};

// Добавляем кнопку для построения графика

buttonPlot = new Button

{

Text = "Построить график",

Location = new System.Drawing.Point(200, 20),

Size = new System.Drawing.Size(120, 30)

};

buttonPlot.Click += ButtonPlot\_Click;

Controls.Add(labelStep);

Controls.Add(textBoxStep);

Controls.Add(buttonPlot);

}

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

{

// Получаем значение шага из текстового поля

if (!double.TryParse(textBoxStep.Text, out double h) || h <= 0)

{

MessageBox.Show("Введите корректное значение шага h (положительное число).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Очищаем предыдущие данные

chart.Series["Series1"].Points.Clear();

// Генерируем данные и добавляем их на график

for (double x = 0; x <= 10; x += h)

{

double y = Math.Exp(x);

chart.Series["Series1"].Points.AddXY(x, y);

}

}

}

}

Таблица 23.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1 |  |

Анализ результатов:

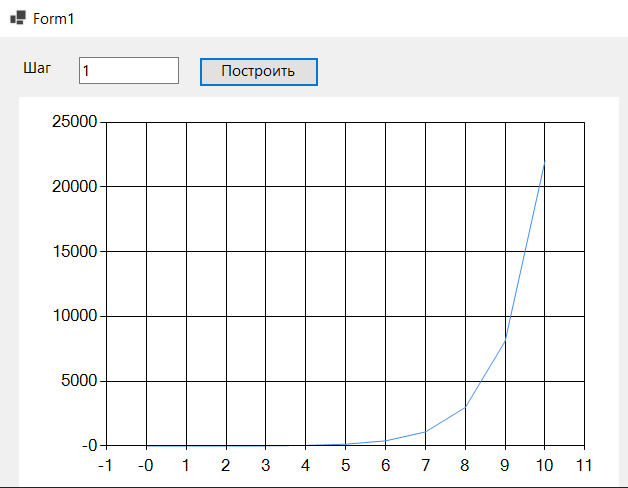


Рисунок 23.2 – Результат работы программы

Задание 3. Построить изображение при помощи фигур. 13. Апельсин

Листинг программы:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WinFormsApp3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

this.Paint += new PaintEventHandler(this.Form1\_Paint);

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

DrawOrange(e.Graphics);

}

private void DrawOrange(Graphics g)

{

// Задаем цвета

Color orangeColor = Color.Orange;

Color darkOrangeColor = Color.DarkOrange;

Color leafGreenColor = Color.Green;

// Создаем кисти

SolidBrush orangeBrush = new SolidBrush(orangeColor);

SolidBrush darkOrangeBrush = new SolidBrush(darkOrangeColor);

SolidBrush leafGreenBrush = new SolidBrush(leafGreenColor);

// Создаем ручки

Pen orangePen = new Pen(orangeColor, 2);

Pen darkOrangePen = new Pen(darkOrangeColor, 2);

Pen leafGreenPen = new Pen(leafGreenColor, 2);

// Рисуем тело апельсина

int centerX = this.ClientSize.Width / 2;

int centerY = this.ClientSize.Height / 2;

int radius = 100;

g.FillEllipse(orangeBrush, centerX - radius, centerY - radius, radius \* 2, radius \* 2);

g.DrawEllipse(orangePen, centerX - radius, centerY - radius, radius \* 2, radius \* 2);

// Рисуем листья апельсина

Point[] leaf1 = {

new Point(centerX, centerY - radius),

new Point(centerX + 40, centerY - radius - 40),

new Point(centerX + 20, centerY - radius)

};

g.FillPolygon(leafGreenBrush, leaf1);

g.DrawPolygon(leafGreenPen, leaf1);

Point[] leaf2 = {

new Point(centerX, centerY - radius),

new Point(centerX - 40, centerY - radius - 40),

new Point(centerX - 20, centerY - radius)

};

g.FillPolygon(leafGreenBrush, leaf2);

g.DrawPolygon(leafGreenPen, leaf2);

// Рисуем текстуру апельсина (тени)

g.DrawArc(darkOrangePen, centerX - radius + 10, centerY - radius + 10, radius \* 2 - 20, radius \* 2 - 20, 45, 180);

g.DrawArc(darkOrangePen, centerX - radius + 20, centerY - radius + 20, radius \* 2 - 40, radius \* 2 - 40, 225, 180);

// Освобождаем ресурсы

orangeBrush.Dispose();

darkOrangeBrush.Dispose();

leafGreenBrush.Dispose();

orangePen.Dispose();

darkOrangePen.Dispose();

leafGreenPen.Dispose();

}

}

}

Таблица 23.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:

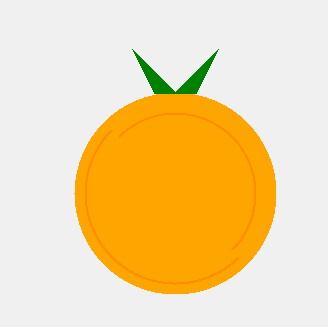


Рисунок 23.3 – Результат работы программы

Задание 4. Написать программу с кнопкой и тремя полями ввода. При нажатии на кнопку программа анализирует содержимое первого поля и динамически пораждает элемент управления. Если в первом поле ввода содержится буква «К», то на форму добавляется кнопка, если буква «П»-поле ввода, если «М»-метка. Во втором и третьем поле находятся координаты левого верхнего угла будущего элемента управления. При создании будущего элемента настроить его свойства (на ваш выбор (цвет, размер и т.п)).При наведении курсора на элемент управления он должен быть удален с формы.

Листинг программы:

using System;

using System.Drawing;

using System.Windows.Forms;

namespace WinFormsApp4

{

public partial class Form1 : Form

{

private TextBox textBoxType;

private TextBox textBoxX;

private TextBox textBoxY;

private Button addButton;

// Делегат для обработки события удаления элемента управления

private delegate void ControlRemovedEventHandler(Control control);

public Form1()

{

InitializeComponent();

InitializeUI(); // Добавляем вызов метода инициализации пользовательского интерфейса

}

private void InitializeComponent()

{

this.textBoxType = new TextBox();

this.textBoxX = new TextBox();

this.textBoxY = new TextBox();

this.addButton = new Button();

this.SuspendLayout();

//

// textBoxType

//

this.textBoxType.Location = new Point(10, 10);

this.textBoxType.Name = "textBoxType";

this.textBoxType.Size = new Size(100, 20);

this.textBoxType.TabIndex = 0;

//

// textBoxX

//

this.textBoxX.Location = new Point(10, 40);

this.textBoxX.Name = "textBoxX";

this.textBoxX.Size = new Size(100, 20);

this.textBoxX.TabIndex = 1;

//

// textBoxY

//

this.textBoxY.Location = new Point(10, 70);

this.textBoxY.Name = "textBoxY";

this.textBoxY.Size = new Size(100, 20);

this.textBoxY.TabIndex = 2;

//

// addButton

//

this.addButton.Location = new Point(10, 100);

this.addButton.Name = "addButton";

this.addButton.Size = new Size(75, 23);

this.addButton.TabIndex = 3;

this.addButton.Text = "Add Control";

this.addButton.UseVisualStyleBackColor = true;

this.addButton.Click += new EventHandler(this.AddButton\_Click);

//

// Form1

//

this.AutoScaleDimensions = new SizeF(6F, 13F);

this.AutoScaleMode = AutoScaleMode.Font;

this.ClientSize = new Size(284, 261);

this.Controls.Add(this.addButton);

this.Controls.Add(this.textBoxY);

this.Controls.Add(this.textBoxX);

this.Controls.Add(this.textBoxType);

this.Name = "Form1";

this.Text = "Dynamic Controls";

this.ResumeLayout(false);

this.PerformLayout();

}

private void InitializeUI()

{

// Добавляем обработчик события MouseHover для всех элементов управления на форме

foreach (Control control in Controls)

{

control.MouseHover += Control\_MouseHover;

}

}

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

{

// Получаем значения из текстовых полей

string controlType = textBoxType.Text;

int x, y;

// Проверяем корректность ввода координат

if (!int.TryParse(textBoxX.Text, out x) || !int.TryParse(textBoxY.Text, out y))

{

MessageBox.Show("Please enter valid integer values for X and Y coordinates.");

return;

}

// Создаем новый элемент управления в зависимости от типа

switch (controlType)

{

case "К":

Button button = new Button();

button.Text = "New Button";

button.Location = new Point(x, y);

Controls.Add(button);

// Добавляем обработчик события для новой кнопки

button.MouseHover += (s, args) => RemoveControl((Control)s);

break;

case "П":

TextBox textBox = new TextBox();

textBox.Text = "New TextBox";

textBox.Location = new Point(x, y);

Controls.Add(textBox);

// Добавляем обработчик события для нового текстового поля

textBox.MouseHover += (s, args) => RemoveControl((Control)s);

break;

case "М":

Label label = new Label();

label.Text = "New Label";

label.Location = new Point(x, y);

Controls.Add(label);

// Добавляем обработчик события для новой метки

label.MouseHover += (s, args) => RemoveControl((Control)s);

break;

default:

MessageBox.Show("Invalid control type. Please enter 'К', 'П', or 'М'.");

break;

}

}

// Метод для удаления элемента управления

private void RemoveControl(Control control)

{

Controls.Remove(control);

}

private void Control\_MouseHover(object sender, EventArgs e)

{

// Проверяем, был ли элемент управления добавлен динамически

if (sender is Control dynamicControl && !Controls.Contains(dynamicControl))

{

// Если да, то удаляем его

Controls.Remove(dynamicControl);

}

}

}

}

Таблица 23.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| M, 300, 300 |  |
|  |  |

Анализ результатов:

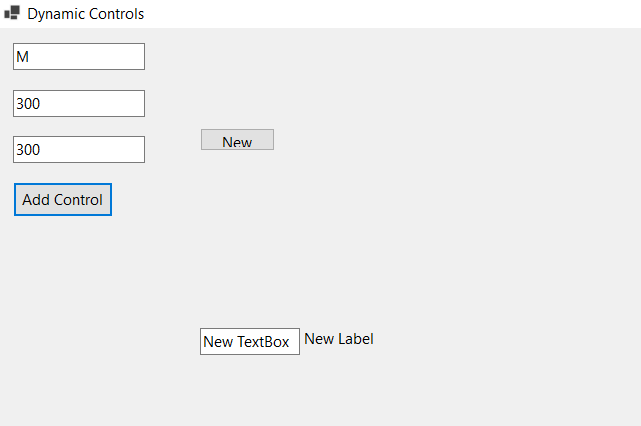


Рисунок 23.4 – Результат работы программы

Задание 5. Разработать Windows-приложение, позволяющее пользователю вводить и выводить анкетные данные служащих.

Листинг программы:

using System.Windows.Forms;

namespace Task4

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

TextBox[] textBoxes = { NameTextBox, SurnameTextBox, MiddleNameTextBox, emailTextBox, phoneBox };

ComboBox[] comboBoxes = { LocationTextBox, operatorBox };

RadioButton[] radioButtons = { MaleRadioButton, FemaleRadioButton };

RadioButton[] workExpRadioButtons = { NoWorkRB, LessThen1RB, From1To5RB, From5To9RB, MoreThen10RB };

CheckBox[] categoryCheckboxes = { categoryA, categoryB, categoryC, categoryD };

RadioButton[] workGraphicRadioButtons = { FullTimeRB, PartTimeRB, WorkAtHomeRB, TimeWorkRB };

NumericUpDown[] numericUpDowns = { FromSalary, ToSalary };

string GetSelectedRadioButtonText(RadioButton[] radioButtons)

{

foreach (RadioButton radioButton in radioButtons)

{

if (radioButton.Checked)

{

return radioButton.Text;

}

}

return "";

}

string GetSelectedCheckBoxText(CheckBox[] checkBoxes)

{

string result = "";

foreach (CheckBox checkBox in checkBoxes)

{

if (checkBox.Checked)

{

result += checkBox.Text;

}

}

return result;

}

string[] textBoxValues = textBoxes.Select(textBox => textBox.Text).ToArray();

string[] comboBoxValues = comboBoxes.Select(comboBox => comboBox.Text).ToArray();

string sex = GetSelectedRadioButtonText(radioButtons);

string dateOfBirth = DateBirtPicker.Text;

string workExp = GetSelectedRadioButtonText(workExpRadioButtons);

if (workExp == "")

{

MessageBox.Show("Не выбран опыт работы!!");

return;

}

string driveCategory = GetSelectedCheckBoxText(categoryCheckboxes);

string workGraphic = GetSelectedRadioButtonText(workGraphicRadioButtons);

if (workGraphic == "")

{

MessageBox.Show("Выберите график работы!!");

return;

}

string salaryFrom = Convert.ToString(FromSalary.Value);

string salaryTo = Convert.ToString(ToSalary.Value);

CVtextBox.AppendText($"Имя: {textBoxValues[0]}\n");

CVtextBox.AppendText($"Фамилия: {textBoxValues[1]}\n");

CVtextBox.AppendText($"Отчество: {textBoxValues[2]}\n");

CVtextBox.AppendText($"Пол: {sex}\n");

CVtextBox.AppendText($"Дата рождения: {dateOfBirth}\n");

CVtextBox.AppendText($"Местоположение: {comboBoxValues[0]}\n");

CVtextBox.AppendText($"Email: {textBoxValues[3]}\n");

CVtextBox.AppendText($"Телефон: {textBoxValues[4]}\n");

CVtextBox.AppendText($"Телефонный оператор: {comboBoxValues[1]}\n");

CVtextBox.AppendText($"Опыт работы: {workExp}\n");

CVtextBox.AppendText($"Собственный автомобиль: {(haveAuto.Checked ? "Да" : "Нет")}\n");

CVtextBox.AppendText($"Водительское удостоверение: {(driveLicense.Checked ? "Да" : "Нет")}\n");

CVtextBox.AppendText($"Категории водительских прав: {driveCategory}\n");

CVtextBox.AppendText($"Зарплата от: {salaryFrom}\n");

CVtextBox.AppendText($"Зарплата до: {salaryTo}\n");

CVtextBox.AppendText($"График работы: {workGraphic}\n");

}

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

{

TextBox[] textBoxes = { NameTextBox, SurnameTextBox, MiddleNameTextBox, emailTextBox, phoneBox };

ComboBox[] comboBoxes = { LocationTextBox, operatorBox };

RadioButton[] radioButtons = { MaleRadioButton, FemaleRadioButton };

RadioButton[] workExpRadioButtons = { NoWorkRB, LessThen1RB, From1To5RB, From5To9RB, MoreThen10RB };

CheckBox[] categoryCheckboxes = { categoryA, categoryB, categoryC, categoryD };

RadioButton[] workGraphicRadioButtons = { FullTimeRB, PartTimeRB, WorkAtHomeRB, TimeWorkRB };

NumericUpDown[] numericUpDowns = { FromSalary, ToSalary };

foreach (TextBox textBox in textBoxes)

{

textBox.Text = "";

}

foreach (ComboBox comboBox in comboBoxes)

{

comboBox.SelectedIndex = -1;

}

foreach (RadioButton radioButton in radioButtons)

{

radioButton.Checked = false;

}

foreach (CheckBox checkBox in categoryCheckboxes)

{

checkBox.Checked = false;

}

foreach (NumericUpDown numericUpDown in numericUpDowns)

{

numericUpDown.Value = numericUpDown.Minimum;

}

CVtextBox.Clear();

}

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

{

Application.Exit();

}

}

}

Таблица 23.5 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Королюк, Артем, Анатольевич, Мужской, 12.06.2006, Гродно, eroshkevich.matvey.mofp@gmail.com, A1, +375445238379, От 1 года до 5 лет, 550, 2550, Полная занятость | Имя: Артем Фамилия: Королюк Отчество: Анатольевич Пол: Мужской Дата рождения: 12 июня 2006 г. Местоположение: Гродно Email: eroshkevich.matvey.mofp@gmail.com Телефон: +375445238379 Телефонный оператор: А1 Опыт работы: От 1 года до 5 лет Собственный автомобиль: Нет Водительское удостоверение: Нет Категории водительских прав: Зарплата от: 550 Зарплата до: 2550 График работы: Полная занятость |

Анализ результатов:

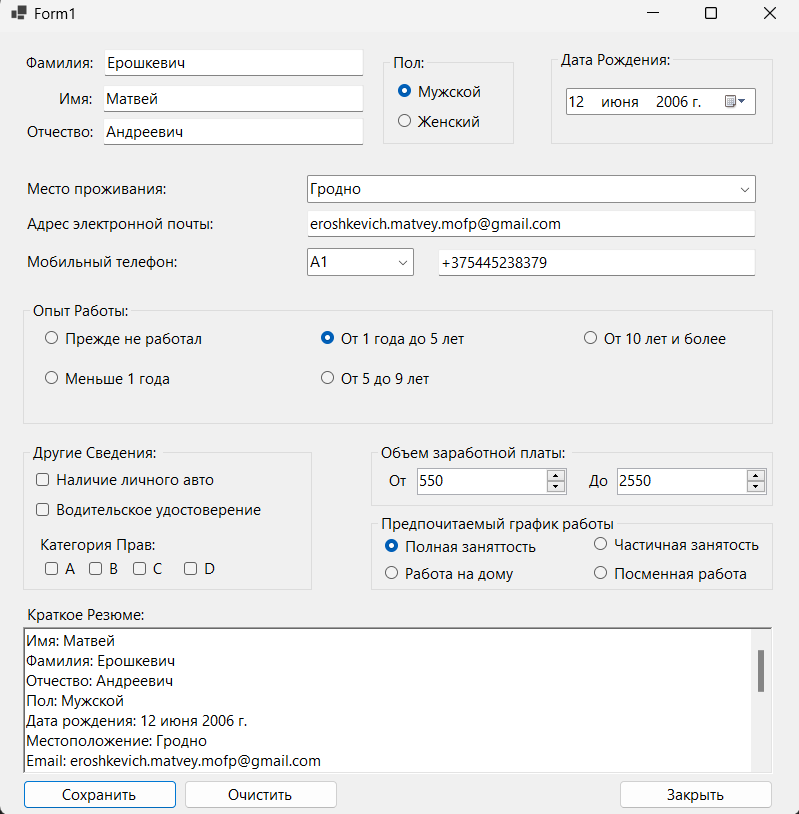


Рисунок 23.5 – Результат работы программы

Задание 6. Создайте пользовательскую форму, которая во время выполнения будет иметь овальное очертание. Данная форма должна содержать функциональность, дающую возможность пользователю закрывать ее во время выполнения. Рекомендация: при разработке формы в виде эллипса используйте следующий код:

Создайте приложение с двумя формами и установите вторую форму как стартовую. Сделайте так, чтобы при запуске стартовая форма разворачивалась до максимальных размеров и содержала функциональность, дающую возможность пользователю открыть первую форму, отображающуюся в виде ромба зеленого цвета с кнопкой (в центре ромба) закрытия формы с надписью GREENPEACE.

Листинг программы:

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

namespace WinFormsApp5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

this.FormBorderStyle = FormBorderStyle.None;

this.BackColor = Color.Violet;

this.StartPosition = FormStartPosition.CenterScreen;

// Сделаем форму овальной

GraphicsPath path = new GraphicsPath();

path.AddEllipse(new Rectangle(0, 0, this.Width, this.Height));

this.Region = new Region(path);

// Добавляем обработчик события клика на форму

this.Click += Form1\_Click;

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

Pen pen = new Pen(Color.Blue, 3);

Rectangle rect = new Rectangle(0, 0, this.Width - 1, this.Height - 1);

g.DrawEllipse(pen, rect);

}

private void InitializeComponent()

{

this.SuspendLayout();

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Name = "Form1";

this.Text = "Oval Form";

this.Paint += new System.Windows.Forms.PaintEventHandler(this.Form1\_Paint);

this.ResumeLayout(false);

}

private void Form1\_Click(object sender, EventArgs e)

{

this.Close(); // Добавляем закрытие формы при клике на неё

}

}

}

Листинг программы:

namespace TwoFormsApp

{

partial class StartForm

{

private System.ComponentModel.IContainer components = null;

private Button openSecondFormButton;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

this.openSecondFormButton = new System.Windows.Forms.Button();

this.SuspendLayout();

//

// openSecondFormButton

//

this.openSecondFormButton.Location = new System.Drawing.Point(12, 12);

this.openSecondFormButton.Name = "openSecondFormButton";

this.openSecondFormButton.Size = new System.Drawing.Size(260, 23);

this.openSecondFormButton.TabIndex = 0;

this.openSecondFormButton.Text = "Open Second Form";

this.openSecondFormButton.UseVisualStyleBackColor = true;

this.openSecondFormButton.Click += new System.EventHandler(this.openSecondFormButton\_Click);

//

// StartForm

//

this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(284, 261);

this.Controls.Add(this.openSecondFormButton);

this.Name = "StartForm";

this.Text = "Start Form";

this.ResumeLayout(false);

}

}

}

Таблица 23.6 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  |  |

Анализ результатов:

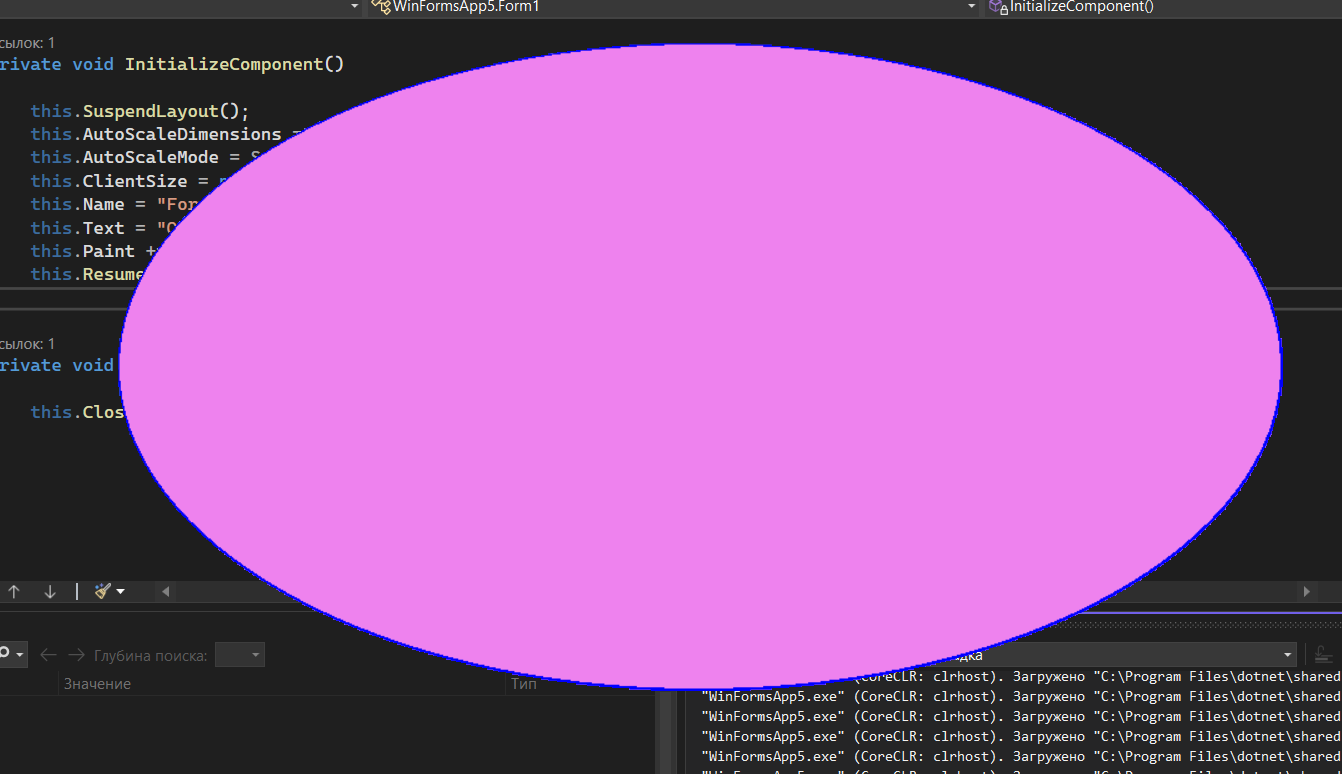


Рисунок 23.6 – Результат работы программы

Анализ результатов:



Рисунок 23.6 – Результат работы программы

Задание 7. Дан массив, содержащий 14 элементов. Все отрицательные элементы заменить на 3. Вывести исходный и полученный массив.

Листинг программы:

using System;

using System.Linq;

using System.Windows.Forms;

namespace ArrayReplacementApp

{

public partial class MainForm : Form

{

private int[] array = { 1, -2, 3, -4, 5, -6, 7, -8, 9, -10, 11, -12, 13, -14 };

public MainForm()

{

InitializeComponent();

DisplayArray(array, originalArrayTextBox);

}

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

{

int[] modifiedArray = (int[])array.Clone();

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

{

if (modifiedArray[i] < 0)

{

modifiedArray[i] = 3;

}

}

DisplayArray(modifiedArray, modifiedArrayTextBox);

}

private void DisplayArray(int[] array, TextBox textBox)

{

textBox.Text = string.Join(", ", array);

}

}

}

Таблица 23.7 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1, -2, 3, -4, 5, -6, 7, -8, 9, -10, 11, -12, 13, -14 | 1, 3, 3, 3, 5, 3, 7, 3, 9, 3, 11, 3, 13, 3 |
|  |  |

Анализ результатов:

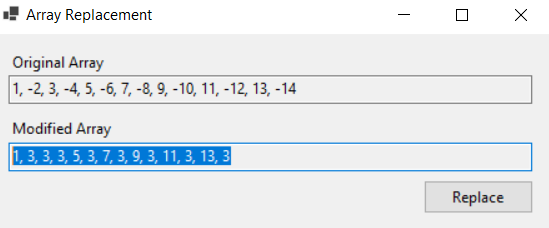


Рисунок 23.7 – Результат работы программы

Задание 8. Вычислить и вывести сумму элементов матрицы A(12,12), расположенных над главной диагональю матрицы.

Листинг программы:

using System;

using System.Windows.Forms;

namespace MatrixSumApp

{

public partial class MainForm : Form

{

private const int Size = 12;

private int[,] matrix = new int[Size, Size];

public MainForm()

{

InitializeComponent();

InitializeMatrix();

DisplayMatrix();

int sum = CalculateSumAboveDiagonal();

sumTextBox.Text = sum.ToString();

}

private void InitializeMatrix()

{

Random rand = new Random();

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

{

for (int j = 0; j < Size; j++)

{

matrix[i, j] = rand.Next(1, 100); // Заполнение матрицы случайными числами от 1 до 99

}

}

}

private void DisplayMatrix()

{

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

{

for (int j = 0; j < Size; j++)

{

matrixTextBox.AppendText(matrix[i, j] + "\t");

}

matrixTextBox.AppendText(Environment.NewLine);

}

}

private int CalculateSumAboveDiagonal()

{

int sum = 0;

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

{

for (int j = i + 1; j < Size; j++)

{

sum += matrix[i, j];

}

}

return sum;

}

}

}

Таблица 23.8 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 69 68 46 33 22 48 10 44 72 67 21 18  73 40 89 96 41 88 65 18 17 53 38 92  24 65 93 7 22 79 35 47 91 48 67 14  92 7 45 55 53 81 65 40 92 1 89 66  83 30 61 72 12 36 60 54 13 37 44 89  96 98 67 7 17 46 22 22 44 61 86 25  32 22 95 68 70 27 68 42 76 23 28 10  32 64 3 25 15 2 19 99 95 31 43 45  53 53 42 12 82 16 5 89 74 59 22 82  45 45 75 80 17 11 77 17 81 86 67 27  72 45 14 98 9 45 66 45 10 30 8 68  68 17 12 20 76 40 78 10 11 22 65 95 | 3254 |
|  |  |

Анализ результатов:

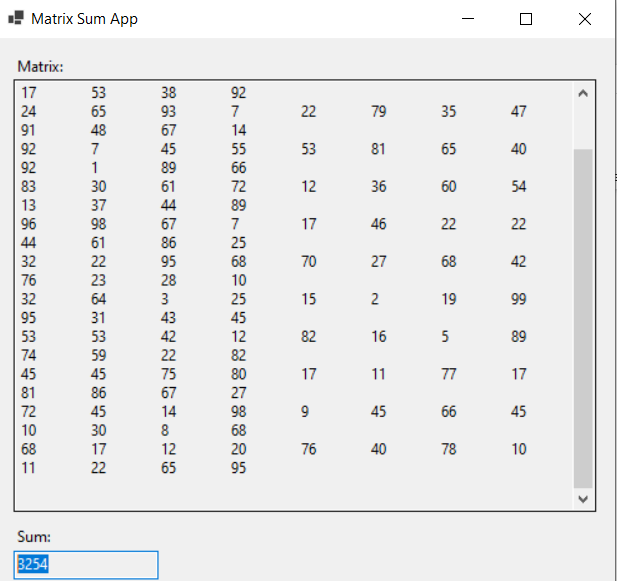


Рисунок 23.8 – Результат работы программы