Министерство образования Республики Беларусь

Оршанский колледж ВГУ имени П.М. Машерова

Отчет

По модулю №5

« Графический интерфейс»

по учебной практике по программированию

Выполнил учащийся Войтеховская К.В.

группа 3ПОИС23 29.09.2025 г.

Проверил Алейников М.А.

30.09.2025 г.

Орша, 2025

**Ход работы.**

**Выполнение практических заданий.**

**Выполнение задания 1.** Создайте графическое приложение для рисования на холсте с использованием Windows Forms. Реализуйте функции рисования линий, кругов и квадратов.

Листинг 1. Код программы приложения для рисования на холсте

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 \_1

{

public partial class Form1 : Form

{

private string currentTool = "Line"; // Текущий инструмент

private Point startPoint; // Начальная точка рисования

private Point endPoint; // Конечная точка рисования

private bool isDrawing = false; // Флаг рисования

private Color currentColor = Color.Black; // Текущий цвет

private List<DrawShape> shapes = new List<DrawShape>(); // Список всех фигур

private List<Point> freehandPoints = new List<Point>(); // Точки для свободного рисования

private bool isFreehandDrawing = false; // Флаг свободного рисования

public Form1()

{

InitializeComponent();

}

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

{

// Настраиваем форму

this.Text = "Простой Paint";

this.Size = new Size(800, 600);

this.BackColor = Color.White;

// Создаем панель инструментов

Panel toolPanel = new Panel();

toolPanel.Dock = DockStyle.Top;

toolPanel.Height = 50;

toolPanel.BackColor = Color.LightGray;

// Кнопка для свободного рисования

Button freehandButton = new Button();

freehandButton.Text = "Карандаш";

freehandButton.Location = new Point(10, 10);

freehandButton.Size = new Size(80, 30);

freehandButton.Click += (s, ev) => { currentTool = "Freehand"; };

// Кнопка для рисования линий

Button lineButton = new Button();

lineButton.Text = "Линия";

lineButton.Location = new Point(100, 10);

lineButton.Size = new Size(80, 30);

lineButton.Click += (s, ev) => { currentTool = "Line"; };

// Кнопка для рисования кругов

Button circleButton = new Button();

circleButton.Text = "Круг";

circleButton.Location = new Point(190, 10);

circleButton.Size = new Size(80, 30);

circleButton.Click += (s, ev) => { currentTool = "Circle"; };

// Кнопка для рисования прямоугольников

Button rectButton = new Button();

rectButton.Text = "Прямоугольник";

rectButton.Location = new Point(280, 10);

rectButton.Size = new Size(100, 30);

rectButton.Click += (s, ev) => { currentTool = "Rectangle"; };

// Кнопка очистки

Button clearButton = new Button();

clearButton.Text = "Очистить";

clearButton.Location = new Point(390, 10);

clearButton.Size = new Size(80, 30);

clearButton.Click += (s, ev) => { shapes.Clear(); freehandPoints.Clear(); this.Invalidate(); };

// Кнопка выбора цвета

Button colorButton = new Button();

colorButton.Text = "Цвет";

colorButton.Location = new Point(480, 10);

colorButton.Size = new Size(80, 30);

colorButton.Click += ColorButton\_Click;

// Добавляем кнопки на панель инструментов

toolPanel.Controls.Add(freehandButton);

toolPanel.Controls.Add(lineButton);

toolPanel.Controls.Add(circleButton);

toolPanel.Controls.Add(rectButton);

toolPanel.Controls.Add(clearButton);

toolPanel.Controls.Add(colorButton);

// Добавляем панель инструментов на форму

this.Controls.Add(toolPanel);

// Настраиваем события мыши

this.MouseDown += Form1\_MouseDown;

this.MouseMove += Form1\_MouseMove;

this.MouseUp += Form1\_MouseUp;

this.Paint += Form1\_Paint;

}

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

isDrawing = true;

startPoint = e.Location;

endPoint = e.Location;

if (currentTool == "Freehand")

{

isFreehandDrawing = true;

freehandPoints.Clear();

freehandPoints.Add(e.Location);

}

}

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

if (isDrawing)

{

if (currentTool == "Freehand" && isFreehandDrawing)

{

// Добавляем точки для свободного рисования

freehandPoints.Add(e.Location);

}

else

{

// Для фигур - просто обновляем конечную точку

endPoint = e.Location;

}

this.Invalidate(); // Перерисовываем форму

}

}

private void Form1\_MouseUp(object sender, MouseEventArgs e)

{

if (isDrawing)

{

isDrawing = false;

endPoint = e.Location;

if (currentTool == "Freehand")

{

isFreehandDrawing = false;

// Сохраняем свободный рисунок как отдельную фигуру

if (freehandPoints.Count > 1)

{

DrawShape newShape = new DrawShape

{

Type = "Freehand",

StartPoint = startPoint,

EndPoint = endPoint,

Color = currentColor,

Points = new List<Point>(freehandPoints) // Сохраняем все точки

};

shapes.Add(newShape);

}

freehandPoints.Clear();

}

else

{

// Создаем новую фигуру и добавляем в список

DrawShape newShape = new DrawShape

{

Type = currentTool,

StartPoint = startPoint,

EndPoint = endPoint,

Color = currentColor

};

shapes.Add(newShape);

}

this.Invalidate();

}

}

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

{

Graphics g = e.Graphics;

g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

// Рисуем все сохраненные фигуры

foreach (DrawShape shape in shapes)

{

DrawShapeOnCanvas(g, shape);

}

// Рисуем текущую фигуру (предпросмотр)

if (isDrawing)

{

if (currentTool == "Freehand" && freehandPoints.Count > 1)

{

// Рисуем текущий свободный рисунок

using (Pen pen = new Pen(Color.FromArgb(128, currentColor), 2))

{

for (int i = 1; i < freehandPoints.Count; i++)

{

g.DrawLine(pen, freehandPoints[i - 1], freehandPoints[i]);

}

}

}

else

{

DrawShape currentShape = new DrawShape

{

Type = currentTool,

StartPoint = startPoint,

EndPoint = endPoint,

Color = Color.FromArgb(128, currentColor) // Полупрозрачный цвет для предпросмотра

};

DrawShapeOnCanvas(g, currentShape);

}

}

}

private void DrawShapeOnCanvas(Graphics g, DrawShape shape)

{

using (Pen pen = new Pen(shape.Color, 2))

{

if (shape.Type == "Freehand" && shape.Points != null && shape.Points.Count > 1)

{

// Рисуем свободную линию по точкам

for (int i = 1; i < shape.Points.Count; i++)

{

g.DrawLine(pen, shape.Points[i - 1], shape.Points[i]);

}

}

else

{

int x = Math.Min(shape.StartPoint.X, shape.EndPoint.X);

int y = Math.Min(shape.StartPoint.Y, shape.EndPoint.Y);

int width = Math.Abs(shape.EndPoint.X - shape.StartPoint.X);

int height = Math.Abs(shape.EndPoint.Y - shape.StartPoint.Y);

switch (shape.Type)

{

case "Line":

g.DrawLine(pen, shape.StartPoint, shape.EndPoint);

break;

case "Circle":

g.DrawEllipse(pen, x, y, width, height);

break;

case "Rectangle":

g.DrawRectangle(pen, x, y, width, height);

break;

}

}

}

}

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

{

ColorDialog colorDialog = new ColorDialog();

colorDialog.Color = currentColor;

if (colorDialog.ShowDialog() == DialogResult.OK)

{

currentColor = colorDialog.Color;

}

}

}

// Класс для хранения информации о фигуре

public class DrawShape

{

public string Type { get; set; } // Line, Circle, Rectangle, Freehand

public Point StartPoint { get; set; }

public Point EndPoint { get; set; }

public Color Color { get; set; }

public List<Point> Points { get; set; } // Для свободного рисования

}

}

На рисунке 1 представлен результат выполнения программы.

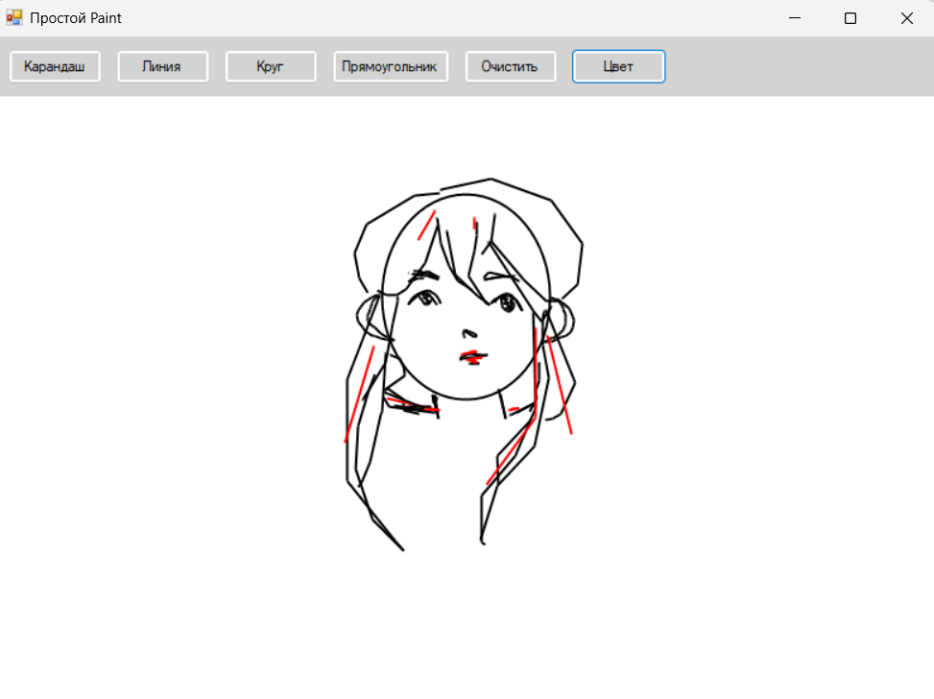


Рисунок 1 – Рисование

**Выполнение задания 2.** Разработайте текстовый редактор с возможностью открытия и сохранения текстовых файлов.

Листинг 2. Код текстового редактора

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;

using System.IO;

namespace \_2

{

public partial class Form1 : Form

{

private string currentFile = ""; // Текущий открытый файл

public Form1()

{

InitializeComponent();

}

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

{

// Настраиваем форму

this.Text = "Простой текстовый редактор";

this.Size = new Size(700, 500);

// Создаем меню

MenuStrip menuStrip = new MenuStrip();

// Меню "Файл"

ToolStripMenuItem fileMenu = new ToolStripMenuItem("Файл");

ToolStripMenuItem newItem = new ToolStripMenuItem("Новый");

newItem.Click += NewItem\_Click;

ToolStripMenuItem openItem = new ToolStripMenuItem("Открыть");

openItem.Click += OpenItem\_Click;

ToolStripMenuItem saveItem = new ToolStripMenuItem("Сохранить");

saveItem.Click += SaveItem\_Click;

ToolStripMenuItem saveAsItem = new ToolStripMenuItem("Сохранить как");

saveAsItem.Click += SaveAsItem\_Click;

ToolStripMenuItem exitItem = new ToolStripMenuItem("Выход");

exitItem.Click += ExitItem\_Click;

fileMenu.DropDownItems.Add(newItem);

fileMenu.DropDownItems.Add(openItem);

fileMenu.DropDownItems.Add(saveItem);

fileMenu.DropDownItems.Add(saveAsItem);

fileMenu.DropDownItems.Add(new ToolStripSeparator());

fileMenu.DropDownItems.Add(exitItem);

menuStrip.Items.Add(fileMenu);

// Добавляем меню на форму

this.Controls.Add(menuStrip);

this.MainMenuStrip = menuStrip;

// Создаем текстовое поле

TextBox textBox = new TextBox();

textBox.Name = "textBox1";

textBox.Multiline = true;

textBox.ScrollBars = ScrollBars.Both;

textBox.Dock = DockStyle.Fill;

textBox.Font = new Font("Arial", 11);

textBox.AcceptsTab = true;

// Добавляем текстовое поле на форму

this.Controls.Add(textBox);

// Устанавливаем порядок отображения

textBox.BringToFront();

}

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

{

TextBox textBox = (TextBox)this.Controls["textBox1"];

textBox.Text = "";

currentFile = "";

this.Text = "Простой текстовый редактор - Новый файл";

}

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

{

OpenFileDialog openDialog = new OpenFileDialog();

openDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

openDialog.Title = "Открыть текстовый файл";

if (openDialog.ShowDialog() == DialogResult.OK)

{

try

{

TextBox textBox = (TextBox)this.Controls["textBox1"];

textBox.Text = File.ReadAllText(openDialog.FileName, Encoding.UTF8);

currentFile = openDialog.FileName;

this.Text = "Простой текстовый редактор - " + Path.GetFileName(currentFile);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при открытии файла: " + ex.Message, "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

if (string.IsNullOrEmpty(currentFile))

{

SaveAsItem\_Click(sender, e);

}

else

{

SaveFile(currentFile);

}

}

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

{

SaveFileDialog saveDialog = new SaveFileDialog();

saveDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";

saveDialog.Title = "Сохранить файл";

if (saveDialog.ShowDialog() == DialogResult.OK)

{

currentFile = saveDialog.FileName;

SaveFile(currentFile);

}

}

private void SaveFile(string filename)

{

try

{

TextBox textBox = (TextBox)this.Controls["textBox1"];

File.WriteAllText(filename, textBox.Text, Encoding.UTF8);

this.Text = "Простой текстовый редактор - " + Path.GetFileName(filename);

MessageBox.Show("Файл успешно сохранен!", "Сохранение",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при сохранении файла: " + ex.Message, "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

}

}

На рисунке 2 представлен результат выполнения программы.

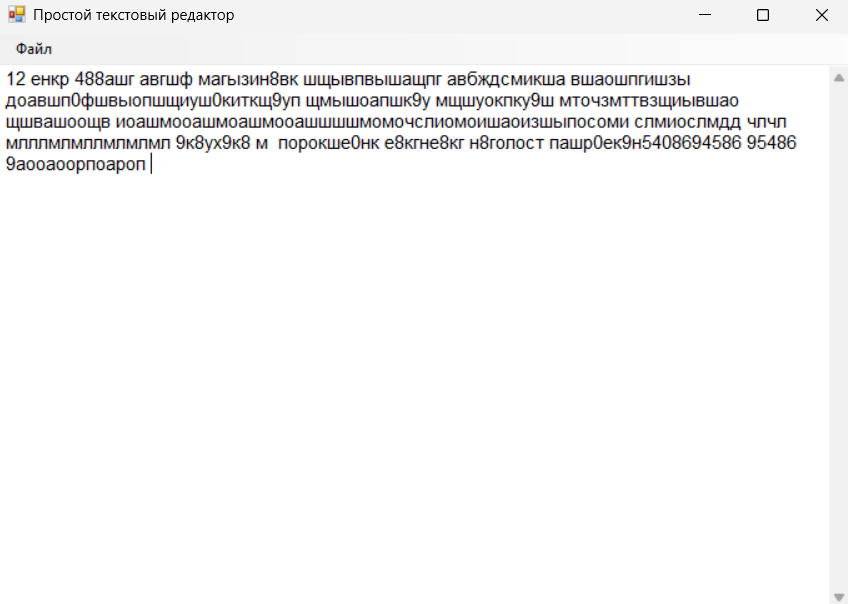


Рисунок 2 – Текстовый редактор

На рисунке 3 представлены пункты меню управления файлами.

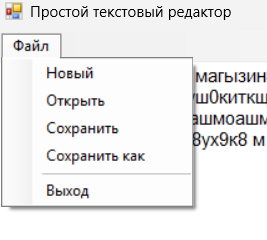


Рисунок 3 – Пункты меню

На рисунке 4 представлено сохранение файлов.

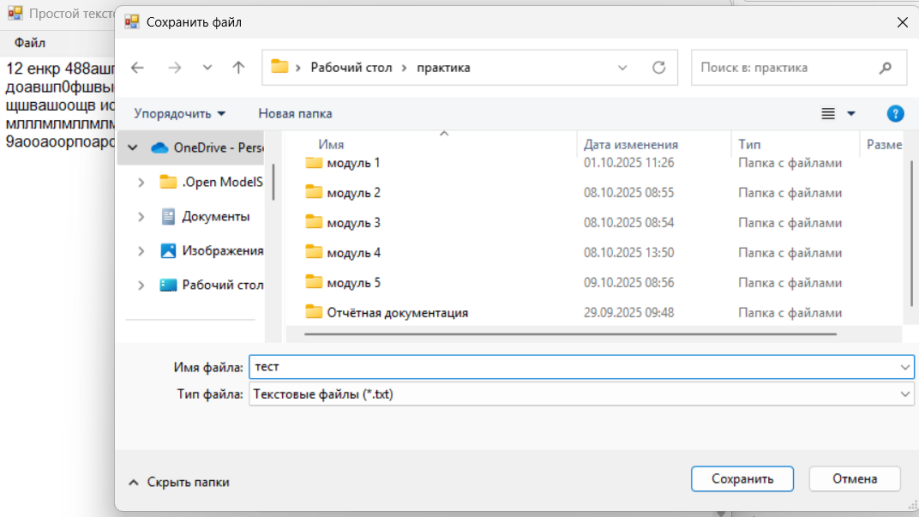


Рисунок 4 – Сохранение файлов

**Выполнение задания 3.** Создайте приложение для учёта задач с использованием Windows Forms. Пользователь должен иметь возможность добавлять, удалять и отмечать задачи как выполненные.

Листинг 3. Код программы для управления задачами

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

оusing System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace \_3

{

public partial class Form1 : Form

{

private ListBox taskListBox;

private TextBox taskTextBox;

private List<TaskItem> tasks = new List<TaskItem>();

public Form1()

{

InitializeComponent();

}

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

{

// Настраиваем форму

this.Text = "Менеджер задач";

this.Size = new Size(500, 400);

this.BackColor = Color.White;

// Заголовок

Label titleLabel = new Label();

titleLabel.Text = "Список задач";

titleLabel.Font = new Font("Arial", 16, FontStyle.Bold);

titleLabel.Location = new Point(20, 20);

titleLabel.Size = new Size(200, 30);

this.Controls.Add(titleLabel);

// Поле для ввода новой задачи

taskTextBox = new TextBox();

taskTextBox.Location = new Point(20, 60);

taskTextBox.Size = new Size(300, 20);

taskTextBox.Font = new Font("Arial", 11);

taskTextBox.Text = "Введите новую задачу...";

taskTextBox.ForeColor = Color.Gray;

taskTextBox.GotFocus += TaskTextBox\_GotFocus;

taskTextBox.LostFocus += TaskTextBox\_LostFocus;

this.Controls.Add(taskTextBox);

// Кнопка добавления задачи

Button addButton = new Button();

addButton.Text = "Добавить";

addButton.Location = new Point(330, 58);

addButton.Size = new Size(80, 25);

addButton.Click += AddButton\_Click;

this.Controls.Add(addButton);

// Enter тоже добавляет задачу

taskTextBox.KeyDown += TaskTextBox\_KeyDown;

// Список задач

taskListBox = new ListBox();

taskListBox.Location = new Point(20, 100);

taskListBox.Size = new Size(390, 200);

taskListBox.Font = new Font("Arial", 10);

taskListBox.SelectionMode = SelectionMode.MultiExtended;

this.Controls.Add(taskListBox);

// Панель с кнопками управления

Panel controlPanel = new Panel();

controlPanel.Location = new Point(20, 310);

controlPanel.Size = new Size(390, 40);

// Кнопка отметки выполнения

Button completeButton = new Button();

completeButton.Text = "Выполнено";

completeButton.Location = new Point(0, 0);

completeButton.Size = new Size(80, 30);

completeButton.Click += CompleteButton\_Click;

controlPanel.Controls.Add(completeButton);

// Кнопка удаления

Button deleteButton = new Button();

deleteButton.Text = "Удалить";

deleteButton.Location = new Point(90, 0);

deleteButton.Size = new Size(80, 30);

deleteButton.Click += DeleteButton\_Click;

controlPanel.Controls.Add(deleteButton);

// Кнопка очистки всех задач

Button clearAllButton = new Button();

clearAllButton.Text = "Очистить все";

clearAllButton.Location = new Point(180, 0);

clearAllButton.Size = new Size(80, 30);

clearAllButton.Click += ClearAllButton\_Click;

controlPanel.Controls.Add(clearAllButton);

// Кнопка сброса отметок

Button resetButton = new Button();

resetButton.Text = "Сбросить";

resetButton.Location = new Point(270, 0);

resetButton.Size = new Size(80, 30);

resetButton.Click += ResetButton\_Click;

controlPanel.Controls.Add(resetButton);

this.Controls.Add(controlPanel);

// Статистика

Label statsLabel = new Label();

statsLabel.Name = "statsLabel";

statsLabel.Location = new Point(20, 360);

statsLabel.Size = new Size(300, 20);

statsLabel.Font = new Font("Arial", 9);

this.Controls.Add(statsLabel);

UpdateStatistics();

}

private void TaskTextBox\_GotFocus(object sender, EventArgs e)

{

// При получении фокуса очищаем текст подсказки

if (taskTextBox.Text == "Введите новую задачу...")

{

taskTextBox.Text = "";

taskTextBox.ForeColor = Color.Black;

}

}

private void TaskTextBox\_LostFocus(object sender, EventArgs e)

{

// При потере фокуса показываем подсказку если поле пустое

if (string.IsNullOrWhiteSpace(taskTextBox.Text))

{

taskTextBox.Text = "Введите новую задачу...";

taskTextBox.ForeColor = Color.Gray;

}

}

private void TaskTextBox\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

AddButton\_Click(sender, e);

}

}

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

{

string taskText = taskTextBox.Text.Trim();

// Игнорируем текст подсказки

if (taskText == "Введите новую задачу...")

{

taskText = "";

}

if (!string.IsNullOrEmpty(taskText))

{

TaskItem newTask = new TaskItem

{

Text = taskText,

IsCompleted = false,

CreatedDate = DateTime.Now

};

tasks.Add(newTask);

taskListBox.Items.Add(newTask.DisplayText);

taskTextBox.Text = "";

taskTextBox.ForeColor = Color.Black;

taskTextBox.Focus();

UpdateStatistics();

}

else

{

MessageBox.Show("Введите текст задачи!", "Внимание",

MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

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

{

if (taskListBox.SelectedIndex >= 0)

{

// Создаем копию выбранных индексов

List<int> selectedIndices = new List<int>();

foreach (int index in taskListBox.SelectedIndices)

{

selectedIndices.Add(index);

}

foreach (int index in selectedIndices)

{

if (index < tasks.Count)

{

tasks[index].IsCompleted = true;

taskListBox.Items[index] = tasks[index].DisplayText;

}

}

UpdateStatistics();

}

else

{

MessageBox.Show("Выберите задачу для отметки!", "Внимание",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

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

{

if (taskListBox.SelectedIndex >= 0)

{

// Создаем копию выбранных индексов в обратном порядке

List<int> selectedIndices = new List<int>();

foreach (int index in taskListBox.SelectedIndices)

{

selectedIndices.Add(index);

}

selectedIndices.Sort((a, b) => b.CompareTo(a));

foreach (int index in selectedIndices)

{

if (index < tasks.Count)

{

tasks.RemoveAt(index);

taskListBox.Items.RemoveAt(index);

}

}

UpdateStatistics();

}

else

{

MessageBox.Show("Выберите задачу для удаления!", "Внимание",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

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

{

if (tasks.Count > 0)

{

DialogResult result = MessageBox.Show("Удалить все задачи?", "Подтверждение",

MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

tasks.Clear();

taskListBox.Items.Clear();

UpdateStatistics();

}

}

else

{

MessageBox.Show("Список задач пуст!", "Информация",

MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

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

{

if (tasks.Count > 0)

{

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

{

tasks[i].IsCompleted = false;

taskListBox.Items[i] = tasks[i].DisplayText;

}

UpdateStatistics();

}

}

private void UpdateStatistics()

{

int totalTasks = tasks.Count;

int completedTasks = tasks.Count(t => t.IsCompleted);

int pendingTasks = totalTasks - completedTasks;

// Находим Label по имени

Control[] foundControls = this.Controls.Find("statsLabel", true);

if (foundControls.Length > 0 && foundControls[0] is Label statsLabel)

{

statsLabel.Text = $"Всего задач: {totalTasks} | Выполнено: {completedTasks} | Осталось: {pendingTasks}";

}

}

}

// Класс для хранения информации о задаче

public class TaskItem

{

public string Text { get; set; }

public bool IsCompleted { get; set; }

public DateTime CreatedDate { get; set; }

public string DisplayText

{

get

{

string status = IsCompleted ? "✓ ВЫПОЛНЕНО" : "● В РАБОТЕ";

string date = CreatedDate.ToString("dd.MM HH:mm");

return $"{status} | {date} | {Text}";

}

}

}

}

На рисунке 5 представлен результат выполнения программы.

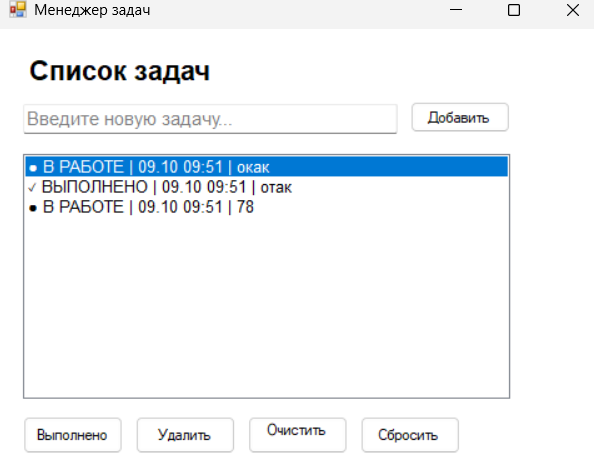


Рисунок 5 – Управление задачи

**Выполнение задания 4.** Реализуйте приложение для просмотра изображений. Пользователь должен иметь возможность выбирать изображение для просмотра и масштабировать его.

Листинг 4. Код программы просмотра изображений

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;

using System.IO;

namespace \_4

{

public partial class Form1 : Form

{

private PictureBox pictureBox;

private float scale = 1.0f;

private string currentImagePath = "";

private Image originalImage;

private ToolStripLabel scaleLabel;

public Form1()

{

InitializeComponent();

}

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

{

// Настраиваем форму

this.Text = "Просмотр изображений";

this.Size = new Size(800, 600);

this.BackColor = Color.White;

// Создаем меню

MenuStrip menuStrip = new MenuStrip();

// Меню "Файл"

ToolStripMenuItem fileMenu = new ToolStripMenuItem("Файл");

ToolStripMenuItem openItem = new ToolStripMenuItem("Открыть");

openItem.Click += OpenItem\_Click;

ToolStripMenuItem exitItem = new ToolStripMenuItem("Выход");

exitItem.Click += ExitItem\_Click;

fileMenu.DropDownItems.Add(openItem);

fileMenu.DropDownItems.Add(new ToolStripSeparator());

fileMenu.DropDownItems.Add(exitItem);

// Меню "Вид"

ToolStripMenuItem viewMenu = new ToolStripMenuItem("Вид");

ToolStripMenuItem zoomInItem = new ToolStripMenuItem("Увеличить");

zoomInItem.Click += ZoomInItem\_Click;

ToolStripMenuItem zoomOutItem = new ToolStripMenuItem("Уменьшить");

zoomOutItem.Click += ZoomOutItem\_Click;

ToolStripMenuItem fitToScreenItem = new ToolStripMenuItem("По размеру экрана");

fitToScreenItem.Click += FitToScreenItem\_Click; ToolStripMenuItem originalSizeItem = new ToolStripMenuItem("Оригинальный размер");

originalSizeItem.Click += OriginalSizeItem\_Click;

viewMenu.DropDownItems.Add(zoomInItem);

viewMenu.DropDownItems.Add(zoomOutItem);

viewMenu.DropDownItems.Add(new ToolStripSeparator());

viewMenu.DropDownItems.Add(fitToScreenItem);

viewMenu.DropDownItems.Add(originalSizeItem);

menuStrip.Items.Add(fileMenu);

menuStrip.Items.Add(viewMenu);

// Создаем панель инструментов

ToolStrip toolStrip = new ToolStrip();

toolStrip.Dock = DockStyle.Top;

// Кнопка открытия

ToolStripButton openButton = new ToolStripButton();

openButton.Text = "Открыть";

openButton.Click += OpenItem\_Click;

// Кнопка увеличения

ToolStripButton zoomInButton = new ToolStripButton();

zoomInButton.Text = "+";

zoomInButton.Click += ZoomInItem\_Click;

// Кнопка уменьшения

ToolStripButton zoomOutButton = new ToolStripButton();

zoomOutButton.Text = "-";

zoomOutButton.Click += ZoomOutItem\_Click;

// Кнопка по размеру экрана

ToolStripButton fitButton = new ToolStripButton();

fitButton.Text = "По размеру";

fitButton.Click += FitToScreenItem\_Click;

// Кнопка оригинального размера

ToolStripButton originalButton = new ToolStripButton();

originalButton.Text = "100%";

originalButton.Click += OriginalSizeItem\_Click;

// Метка масштаба

scaleLabel = new ToolStripLabel();

scaleLabel.Text = "100%";

toolStrip.Items.Add(openButton);

toolStrip.Items.Add(new ToolStripSeparator());

toolStrip.Items.Add(zoomInButton);

toolStrip.Items.Add(zoomOutButton);

toolStrip.Items.Add(fitButton);

toolStrip.Items.Add(originalButton);

toolStrip.Items.Add(new ToolStripSeparator());

toolStrip.Items.Add(scaleLabel);

// Создаем PictureBox для отображения изображения

pictureBox = new PictureBox();

pictureBox.Dock = DockStyle.Fill;

pictureBox.SizeMode = PictureBoxSizeMode.Zoom;

pictureBox.BorderStyle = BorderStyle.FixedSingle;

pictureBox.BackColor = Color.LightGray;

// Добавляем элементы на форму

this.Controls.Add(pictureBox);

this.Controls.Add(toolStrip);

this.Controls.Add(menuStrip);

this.MainMenuStrip = menuStrip;

// Настраиваем события колесика мыши для масштабирования

pictureBox.MouseWheel += PictureBox\_MouseWheel;

}

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

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.Filter = "Изображения|\*.jpg;\*.jpeg;\*.png;\*.bmp;\*.gif|Все файлы|\*.\*";

dialog.Title = "Выберите изображение";

if (dialog.ShowDialog() == DialogResult.OK)

{

try

{

// Освобождаем предыдущее изображение

if (originalImage != null)

{

originalImage.Dispose();

originalImage = null;

}

currentImagePath = dialog.FileName;

originalImage = Image.FromFile(currentImagePath);

pictureBox.Image = originalImage;

scale = 1.0f;

UpdateScaleLabel();

this.Text = "Просмотр изображений - " + Path.GetFileName(currentImagePath);

// Устанавливаем режим масштабирования

pictureBox.SizeMode = PictureBoxSizeMode.Zoom;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка загрузки изображения: " + ex.Message, "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

if (pictureBox.Image != null)

{

scale += 0.1f;

ApplyScale();

}

}

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

{

if (pictureBox.Image != null && scale > 0.1f)

{

scale -= 0.1f;

ApplyScale();

}

}

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

{

if (pictureBox.Image != null)

{

pictureBox.SizeMode = PictureBoxSizeMode.Zoom;

scale = 1.0f;

UpdateScaleLabel();

}

}

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

{

if (pictureBox.Image != null)

{

pictureBox.SizeMode = PictureBoxSizeMode.AutoSize;

scale = 1.0f;

UpdateScaleLabel();

}

}

private void PictureBox\_MouseWheel(object sender, MouseEventArgs e)

{

if (pictureBox.Image != null)

{

if (e.Delta > 0)

{

// Колесико вверх - увеличиваем

scale += 0.1f;

}

else

{

// Колесико вниз - уменьшаем

if (scale > 0.1f)

{

scale -= 0.1f;

}

}

ApplyScale();

}

}

private void ApplyScale()

{

if (pictureBox.Image != null && originalImage != null)

{

// Создаем новое изображение с нужным размером

int newWidth = (int)(originalImage.Width \* scale);

int newHeight = (int)(originalImage.Height \* scale);

// Создаем временное изображение

Bitmap tempImage = new Bitmap(originalImage, newWidth, newHeight);

// Устанавливаем новое изображение

pictureBox.Image = tempImage;

pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;

UpdateScaleLabel();

}

}

private void UpdateScaleLabel()

{

scaleLabel.Text = $"{scale \* 100:0}%";

}

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

{

this.Close();

}

// Освобождаем ресурсы при закрытии формы

protected override void OnFormClosed(FormClosedEventArgs e)

{

base.OnFormClosed(e);

if (originalImage != null)

{

originalImage.Dispose();

}

if (pictureBox.Image != null && pictureBox.Image != originalImage)

{

pictureBox.Image.Dispose();

}

}

}

}

На рисунке 6 представлен результат выполнения программы.

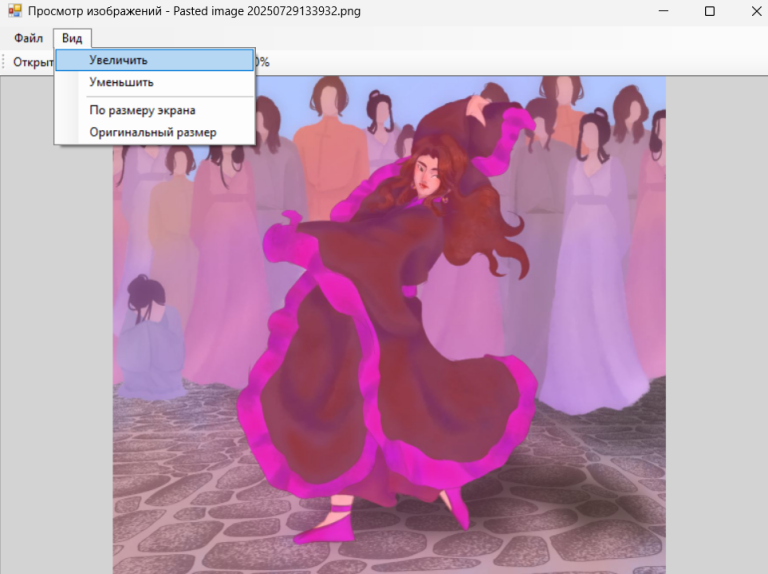


Рисунок 6 – Просмотр приложение

**Выполнение задания 5.** Создайте калькулятор с графическим интерфейсом. Пользователь должен иметь возможность выполнять арифметические операции.

Листинг 5. Код программы калькулятора

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 \_5

{

public partial class Form1 : Form

{

private TextBox display;

private double firstNumber = 0;

private string operation = "";

private bool newOperation = true;

public Form1()

{

InitializeComponent();

}

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

{

// Настраиваем форму

this.Text = "Калькулятор";

this.Size = new Size(300, 400);

this.BackColor = Color.White;

this.FormBorderStyle = FormBorderStyle.FixedDialog;

this.MaximizeBox = false;

// Создаем дисплей

display = new TextBox();

display.Location = new Point(10, 10);

display.Size = new Size(260, 30);

display.Font = new Font("Arial", 14);

display.TextAlign = HorizontalAlignment.Right;

display.ReadOnly = true;

display.Text = "0";

this.Controls.Add(display);

// Массив кнопок калькулятора

string[,] buttonLayout = {

{"C", "±", "%", "÷"},

{"7", "8", "9", "×"},

{"4", "5", "6", "-"},

{"1", "2", "3", "+"},

{"0", "", ".", "="}

};

int startX = 10;

int startY = 50;

int buttonWidth = 60;

int buttonHeight = 50;

int spacing = 5;

// Создаем кнопки

for (int row = 0; row < buttonLayout.GetLength(0); row++)

{

for (int col = 0; col < buttonLayout.GetLength(1); col++)

{

string buttonText = buttonLayout[row, col];

if (!string.IsNullOrEmpty(buttonText))

{

Button button = new Button();

button.Text = buttonText;

button.Size = new Size(buttonWidth, buttonHeight);

button.Location = new Point(

startX + col \* (buttonWidth + spacing),

startY + row \* (buttonHeight + spacing)

);

button.Font = new Font("Arial", 12);

// Назначаем обработчики событий

if (char.IsDigit(buttonText[0]) || buttonText == ".")

{

button.Click += NumberButton\_Click;

}

else if (buttonText == "=")

{

button.Click += EqualsButton\_Click;

}

else if (buttonText == "C")

{

button.Click += ClearButton\_Click;

}

else if (buttonText == "±")

{

button.Click += PlusMinusButton\_Click;

}

else if (buttonText == "%")

{

button.Click += PercentButton\_Click;

}

else

{

button.Click += OperatorButton\_Click;

}

this.Controls.Add(button);

}

}

}

// Делаем кнопку 0 шире

Button zeroButton = this.Controls.OfType<Button>().First(b => b.Text == "0");

zeroButton.Size = new Size(buttonWidth \* 2 + spacing, buttonHeight);

}

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

{

Button button = (Button)sender;

string number = button.Text;

if (display.Text == "0" || newOperation)

{

display.Text = number;

newOperation = false;

}

else

{

// Проверяем, чтобы точка была только одна

if (number == "." && display.Text.Contains("."))

{

return;

}

display.Text += number;

}

}

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

{

Button button = (Button)sender;

if (!newOperation)

{

Calculate();

}

firstNumber = double.Parse(display.Text);

operation = button.Text;

newOperation = true;

}

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

{

if (!newOperation && !string.IsNullOrEmpty(operation))

{

Calculate();

operation = "";

newOperation = true;

}

}

private void Calculate()

{

double secondNumber = double.Parse(display.Text);

double result = 0;

switch (operation)

{

case "+":

result = firstNumber + secondNumber;

break;

case "-":

result = firstNumber - secondNumber;

break;

case "×":

result = firstNumber \* secondNumber;

break;

case "÷":

if (secondNumber != 0)

{

result = firstNumber / secondNumber;

}

else

{

display.Text = "Ошибка";

return;

}

break;

}

// Форматируем результат

display.Text = result.ToString().Replace(",", ".");

if (display.Text.Contains("."))

{

display.Text = display.Text.TrimEnd('0').TrimEnd('.');

if (display.Text == "") display.Text = "0";

}

}

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

{

display.Text = "0";

firstNumber = 0;

operation = "";

newOperation = true;

}

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

{

if (display.Text != "0" && display.Text != "Ошибка")

{

double number = double.Parse(display.Text);

number = -number;

display.Text = number.ToString().Replace(",", ".");

}

}

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

{

if (display.Text != "0" && display.Text != "Ошибка")

{

double number = double.Parse(display.Text);

number = number / 100;

display.Text = number.ToString().Replace(",", ".");

newOperation = true;

}

}

// Обработка нажатия клавиш на клавиатуре

protected override bool ProcessCmdKey(ref Message msg, Keys keyData)

{

switch (keyData)

{

case Keys.D0:

case Keys.NumPad0:

SimulateButtonClick("0");

return true;

case Keys.D1:

case Keys.NumPad1:

SimulateButtonClick("1");

return true;

case Keys.D2:

case Keys.NumPad2:

SimulateButtonClick("2");

return true;

case Keys.D3:

case Keys.NumPad3:

SimulateButtonClick("3");

return true;

case Keys.D4:

case Keys.NumPad4:

SimulateButtonClick("4");

return true;

case Keys.D5:

case Keys.NumPad5:

SimulateButtonClick("5");

return true;

case Keys.D6:

case Keys.NumPad6:

SimulateButtonClick("6");

return true;

case Keys.D7:

case Keys.NumPad7:

SimulateButtonClick("7");

return true;

case Keys.D8:

case Keys.NumPad8:

SimulateButtonClick("8");

return true;

case Keys.D9:

case Keys.NumPad9:

SimulateButtonClick("9");

return true;

case Keys.Add:

SimulateButtonClick("+");

return true;

case Keys.Subtract:

SimulateButtonClick("-");

return true;

case Keys.Multiply:

SimulateButtonClick("×");

return true;

case Keys.Divide:

SimulateButtonClick("÷");

return true;

case Keys.Enter:

SimulateButtonClick("=");

return true;

case Keys.Escape:

SimulateButtonClick("C");

return true;

case Keys.Decimal:

SimulateButtonClick(".");

return true;

}

return base.ProcessCmdKey(ref msg, keyData);

}

private void SimulateButtonClick(string buttonText)

{

var button = this.Controls.OfType<Button>().FirstOrDefault(b => b.Text == buttonText);

if (button != null)

{

button.PerformClick();

}

}

}

}

На рисунке 7 представлен результат выполнения программы.

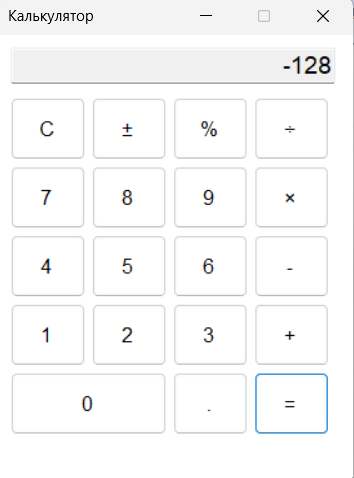


Рисунок 7 – Калькулятор