**Виконав: Мельник Б. В.; КН-922в**

**Лабораторна робота №11**

**Тема:** Створення багатовіконних застосунків. Частина 1.

**Мета роботи:** набути навички у створенні багатовіконних застосунків.

**Індивідуальні завдання**

**Варіант №12**

**Завдання 1**

Використовуючи граф-схему власного варіанту, створити багатовіконний застосунок виклику відповідних форм

|  |  |
| --- | --- |
| № варіанту | Граф схема |
| 1 | 2 |
| 12. |  |

Назви форм повинні частково відповідати назвам з вершин графів (наприклад Form1, Form2, ... чи FormA, FormB, FormC, ...). Форми мають викликатися в модальному режимі (ShowDialog) та мати різнокольорові характеристики фону та тексту. Виклики форм виконувати за допомогою звичайних кнопок, кнопок з панелей ToolStrip, a6o за допомогою команд з MenuStrip чи ContextMenuStrip.

**Текст програми:**

***MainForm.cs:***

using Lab\_11;

using System;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class MainForm : Form

{

public MainForm ()

{

InitializeComponent();

}

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

{

FormG formG = new FormG();

formG.ShowDialog();

}

}

}

***FormG.cs:***

using Lab\_11;

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormG : Form

{

public FormG()

{

InitializeComponent();

// Customize button fonts and colors

buttonOpenFormH.Font = new Font("Arial", 10, FontStyle.Bold);

buttonOpenFormH.ForeColor = Color.Red;

buttonOpenFormI.Font = new Font("Times New Roman", 12, FontStyle.Italic);

buttonOpenFormI.ForeColor = Color.Blue;

buttonOpenFormJ.Font = new Font("Verdana", 10, FontStyle.Underline);

buttonOpenFormJ.ForeColor = Color.Green;

}

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

{

FormH formH = new FormH();

formH.ShowDialog();

}

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

{

FormI formI = new FormI();

formI.ShowDialog();

}

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

{

FormJ formJ = new FormJ();

formJ.ShowDialog();

}

}

}

***FormH.cs:***

using Lab\_11;

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormH : Form

{

public FormH()

{

InitializeComponent();

buttonOpenFormM.Font = new Font("Courier New", 12, FontStyle.Bold);

buttonOpenFormM.ForeColor = Color.Purple;

buttonOpenFormN.Font = new Font("Calibri", 11, FontStyle.Strikeout);

buttonOpenFormN.ForeColor = Color.DarkOrange;

}

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

{

FormM formM = new FormM();

formM.ShowDialog();

}

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

{

FormN formN = new FormN();

formN.ShowDialog();

}

}

}

***FormI.cs:***

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormI : Form

{

public FormI()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Comic Sans MS", 10, FontStyle.Bold | FontStyle.Italic),

ForeColor = Color.DarkCyan,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

}

}

***FormJ.cs:***

using Lab\_11;

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormJ : Form

{

public FormJ()

{

InitializeComponent();

buttonOpenFormK.Font = new Font("Segoe UI", 10, FontStyle.Bold);

buttonOpenFormK.ForeColor = Color.DarkBlue;

buttonOpenFormL.Font = new Font("Georgia", 10, FontStyle.Italic);

buttonOpenFormL.ForeColor = Color.Crimson;

}

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

{

FormK formK = new FormK();

formK.ShowDialog();

}

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

{

FormL formL = new FormL();

formL.ShowDialog();

}

}

}

***FormK.cs:***

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormK : Form

{

public FormK()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Tahoma", 11, FontStyle.Bold),

ForeColor = Color.Navy,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

}

}

***FormL.cs:***

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormL : Form

{

public FormL()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Palatino Linotype", 12, FontStyle.Underline),

ForeColor = Color.Teal,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

}

}

***FormM.cs:***

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormM : Form

{

public FormM()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Arial Narrow", 10, FontStyle.Italic),

ForeColor = Color.Maroon,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

}

}

***FormN.cs:***

using System.Drawing;

using System.Windows.Forms;

namespace Lab\_11

{

public partial class FormN : Form

{

public FormN()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Century Gothic", 10, FontStyle.Strikeout),

ForeColor = Color.DarkMagenta,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

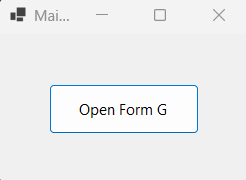
this.Controls.Add(buttonExample);

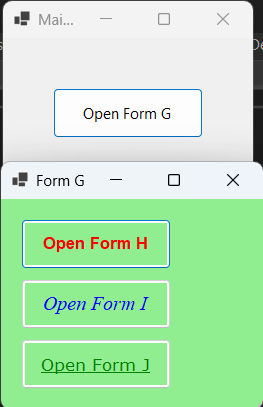
}

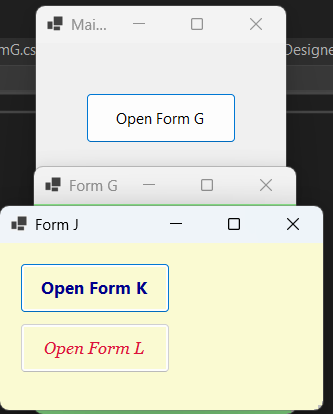
}

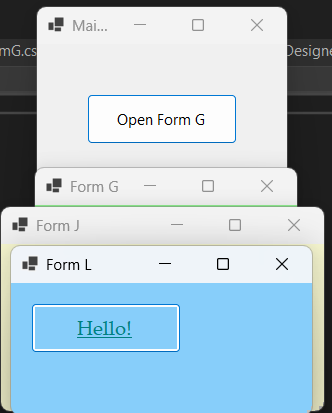
}

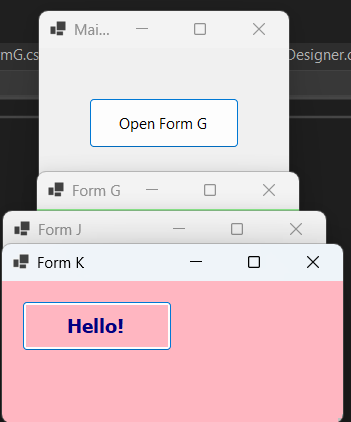
**Результат роботи програми:**

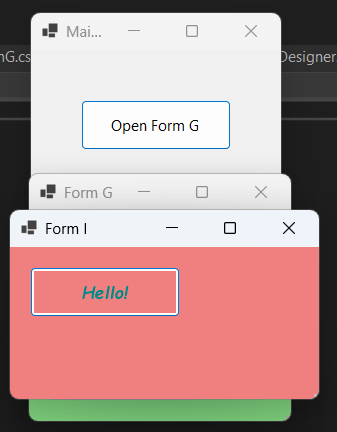


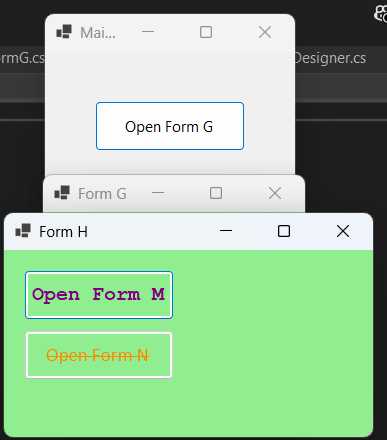


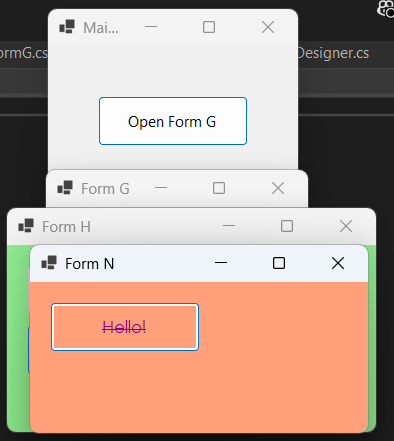


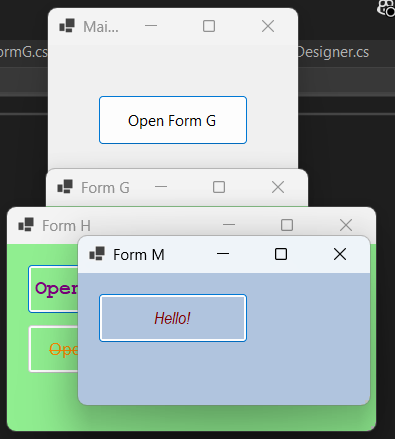












**Алгоритм побудови коду**

1. **Головна форма** (MainForm):
   * Створюється кнопка, яка відкриває наступну форму графа (наприклад, FormG).
   * Кнопка використовує подію Click, яка ініціює метод ShowDialog для відкриття FormG у модальному режимі.
2. **Форма FormG**:
   * Створює кнопки для переходу до FormH, FormI, FormJ.
   * Кожна кнопка налаштована з унікальними кольорами тексту та стилями шрифту, що відображають різні стилі ідентифікації форм.
   * Кожна кнопка викликає відповідну форму (наприклад, FormH) через ShowDialog.
3. **Форми FormH, FormI, FormJ та інші**:
   * Кожна з цих форм містить кнопки для відкриття наступних форм (наприклад, FormM, FormN).
   * Усі кнопки мають стилізовані шрифти, кольори, що відрізняють їх зовнішній вигляд.
   * Кнопки налаштовані на відкриття інших форм відповідно до граф-схеми, також у модальному режимі.
4. **Стилізація елементів**:
   * У кожній формі встановлено різні стилі для шрифтів, кольорів фону та тексту, щоб забезпечити різнобарвність інтерфейсу і відповідність вимогам завдання.

**Функціональність**

* **Перехід між формами**: Основна функція – послідовний перехід між формами, що дозволяє користувачеві відкривати інші форми згідно з графом, заданим у варіанті.
* **Модальний режим**: Усі форми відкриваються в модальному режимі (ShowDialog), що забезпечує фокус користувача на одній формі за раз.
* **Унікальні стилі елементів**: Кожна форма має свої унікальні налаштування шрифтів, кольорів, щоб покращити візуальне сприйняття та відрізнити форми.

**Висновок**

Завдання було успішно виконане, створено багатовіконний застосунок із модальними формами. Використання різнокольорових стилів для кнопок і текстів покращило візуальну структуру застосунку, роблячи його більш зрозумілим для користувача. Алгоритм забезпечує легке розширення коду, дозволяючи додавати нові форми або елементи управління, що спрощує масштабування застосунку.

**Завдання 2\***

Модифікувати минуле завдання, додавши до застосунку можливість обмінюватися інформаціею між формами наступним чином:

* додати до кожної форми компонент(и) label, в яких мае відображатися назва та інформація з форми, яка викликала a6o повернула управління до цісї форми;
* додати до кожної форми компонент textbox, де е можливість друкувати інформацію для викликаючої, a6o повертаючої управління форми;
* під час виклику чи повертання з форм відповідні поля та мітки мають заповнюватися відповідної інформаціею з форм.

**Текст програми:**

***MainForm.cs:***

using Lab\_11\_Task\_2;

using System;

using System.Windows.Forms;

namespace Lab\_11\_Task\_2

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

this.labelInfo.Text = "Welcome!";

}

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

{

FormG formG = new FormG();

// Відправка даних до форми G

formG.CallingFormName = this.Text;

formG.CallingFormInfo = this.textBoxInfo.Text;

formG.ShowDialog();

// Прийняття даних від форми G

this.labelInfo.Text = $"From {formG.Text}: {formG.ReturnedInfo}";

}

}

}

***FormG.cs:***

using Lab\_11\_Task\_2;

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormG : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormG()

{

InitializeComponent();

// Налаштування шрифтів і кольорів кнопок

buttonOpenFormH.Font = new Font("Arial", 10, FontStyle.Bold);

buttonOpenFormH.ForeColor = Color.Red;

buttonOpenFormI.Font = new Font("Times New Roman", 12, FontStyle.Italic);

buttonOpenFormI.ForeColor = Color.Blue;

buttonOpenFormJ.Font = new Font("Verdana", 10, FontStyle.Underline);

buttonOpenFormJ.ForeColor = Color.Green;

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

// Відображення імені та інформації форми виклику

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

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

{

FormH formH = new FormH();

// Передача даних до FormH

formH.CallingFormName = this.Text;

formH.CallingFormInfo = this.textBoxInfo.Text;

formH.ShowDialog();

// Прийняття даних від FormH

this.labelInfo.Text = $"From {formH.Text}: {formH.ReturnedInfo}";

}

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

{

FormI formI = new FormI();

// Передача даних до FormI

formI.CallingFormName = this.Text;

formI.CallingFormInfo = this.textBoxInfo.Text;

formI.ShowDialog();

// Прийняття даних від FormI

this.labelInfo.Text = $"From {formI.Text}: {formI.ReturnedInfo}";

}

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

{

FormJ formJ = new FormJ();

// Передача даних до FormJ

formJ.CallingFormName = this.Text;

formJ.CallingFormInfo = this.textBoxInfo.Text;

formJ.ShowDialog();

// Прийняття даних від FormJ

this.labelInfo.Text = $"From {formJ.Text}: {formJ.ReturnedInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

// Налаштування ReturnedInfo для надсилання назад до форми виклику

ReturnedInfo = textBoxInfo.Text;

base.OnFormClosing(e);

}

}

}

***FormH.cs:***

using Lab\_11\_Task\_2;

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormH : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormH()

{

InitializeComponent();

buttonOpenFormM.Font = new Font("Courier New", 12, FontStyle.Bold);

buttonOpenFormM.ForeColor = Color.Purple;

buttonOpenFormN.Font = new Font("Calibri", 11, FontStyle.Strikeout);

buttonOpenFormN.ForeColor = Color.DarkOrange;

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

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

{

FormM formM = new FormM();

// Передача даних до форми M

formM.CallingFormName = this.Text;

formM.CallingFormInfo = this.textBoxInfo.Text;

formM.ShowDialog();

// Прийняття даних від форми M

this.labelInfo.Text = $"From {formM.Text}: {formM.ReturnedInfo}";

}

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

{

FormN formN = new FormN();

// Передача даних до форми N

formN.CallingFormName = this.Text;

formN.CallingFormInfo = this.textBoxInfo.Text;

formN.ShowDialog();

// Прийняття даних від форми N

this.labelInfo.Text = $"From {formN.Text}: {formN.ReturnedInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

ReturnedInfo = textBoxInfo.Text;

base.OnFormClosing(e);

}

}

}

***FormI.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormI : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormI()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Comic Sans MS", 10, FontStyle.Bold | FontStyle.Italic),

ForeColor = Color.DarkCyan,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

ReturnedInfo = textBoxInfo.Text;

base.OnFormClosing(e);

}

}

}

***FormJ.cs:***

using Lab\_11\_Task\_2;

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormJ : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormJ()

{

InitializeComponent();

buttonOpenFormK.Font = new Font("Segoe UI", 10, FontStyle.Bold);

buttonOpenFormK.ForeColor = Color.DarkBlue;

buttonOpenFormL.Font = new Font("Georgia", 10, FontStyle.Italic);

buttonOpenFormL.ForeColor = Color.Crimson;

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

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

{

FormK formK = new FormK();

// Передача даних до форми K

formK.CallingFormName = this.Text;

formK.CallingFormInfo = this.textBoxInfo.Text;

formK.ShowDialog();

// Прийняття даних від форми K

this.labelInfo.Text = $"From {formK.Text}: {formK.ReturnedInfo}";

}

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

{

FormL formL = new FormL();

// Передача даних до форми L

formL.CallingFormName = this.Text;

formL.CallingFormInfo = this.textBoxInfo.Text;

formL.ShowDialog();

// Прийняття даних від форми L

this.labelInfo.Text = $"From {formL.Text}: {formL.ReturnedInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

ReturnedInfo = textBoxInfo.Text;

base.OnFormClosing(e);

}

}

}

***FormK.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormK : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormK()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Tahoma", 11, FontStyle.Bold),

ForeColor = Color.Navy,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

ReturnedInfo = textBoxInfo.Text;

base.OnFormClosing(e);

}

}

}

***FormL.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormL : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormL()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Palatino Linotype", 12, FontStyle.Underline),

ForeColor = Color.Teal,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

ReturnedInfo = textBoxInfo.Text;

base.OnFormClosing(e);

}

}

}

***FormM.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormM : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormM()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Arial Narrow", 10, FontStyle.Italic),

ForeColor = Color.Maroon,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

ReturnedInfo = textBoxInfo.Text;

base.OnFormClosing(e);

}

}

}

***FormN.cs:***

using System;

using System.Drawing;

using System.Windows.Forms;

using static System.Runtime.InteropServices.Marshalling.IIUnknownCacheStrategy;

namespace Lab\_11\_Task\_2

{

public partial class FormN : Form

{

public string CallingFormName { get; set; }

public string CallingFormInfo { get; set; }

public string ReturnedInfo { get; set; }

public FormN()

{

InitializeComponent();

Button buttonExample = new Button

{

Text = "Hello!",

Font = new Font("Century Gothic", 10, FontStyle.Strikeout),

ForeColor = Color.DarkMagenta,

Location = new Point(20, 20),

Size = new Size(150, 50)

};

this.Controls.Add(buttonExample);

}

protected override void OnLoad(EventArgs e)

{

base.OnLoad(e);

labelInfo.Text = $"Called from {CallingFormName}: {CallingFormInfo}";

}

protected override void OnFormClosing(FormClosingEventArgs e)

{

ReturnedInfo = textBoxInfo.Text;

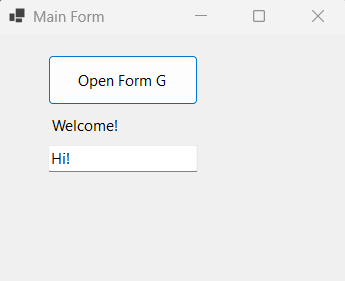
base.OnFormClosing(e);

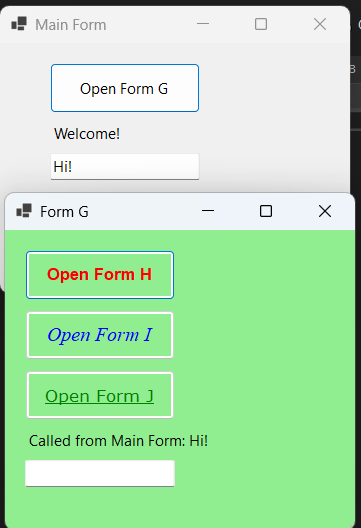
}

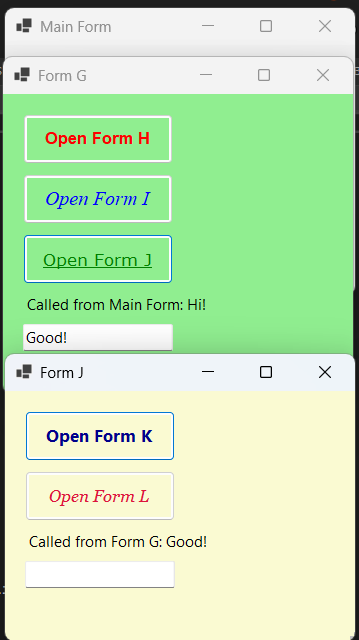
}

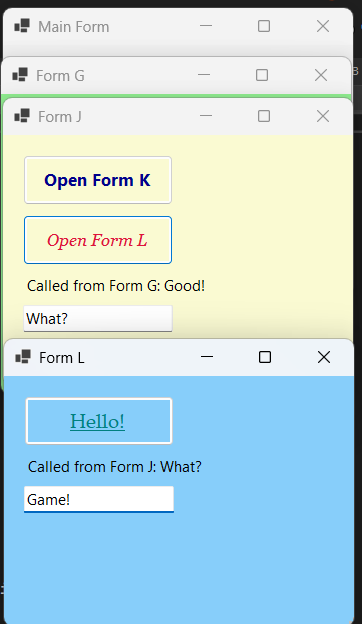
}

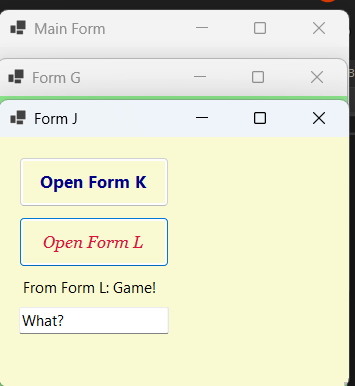
**Результат роботи програми:**

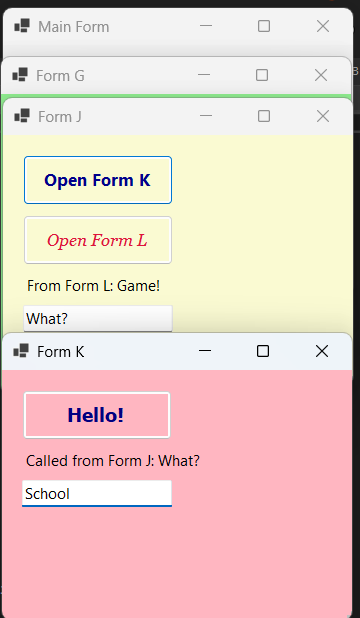


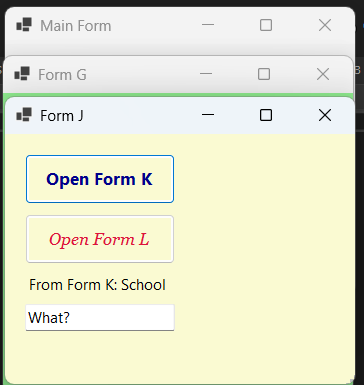


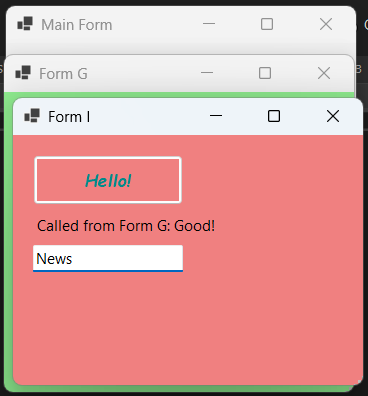


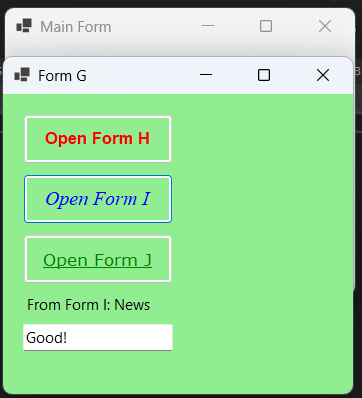


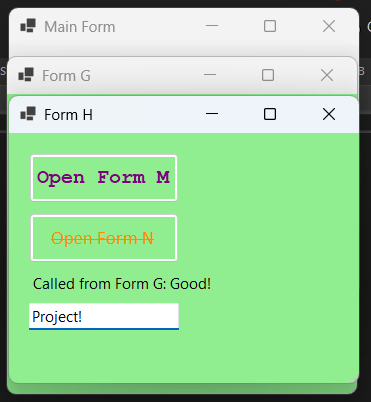


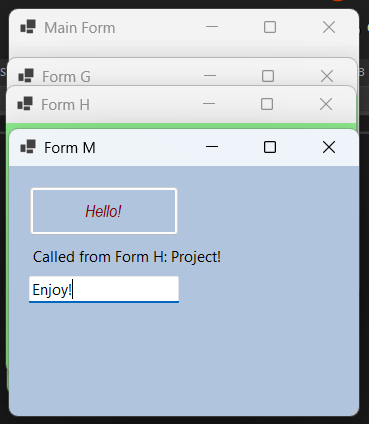


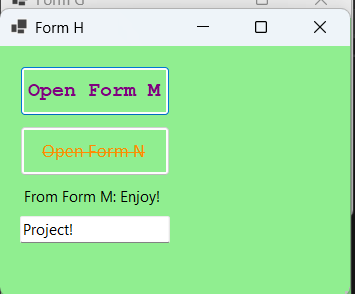


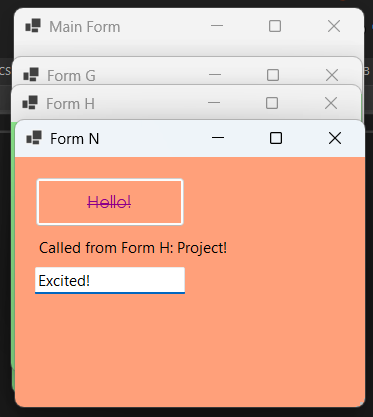


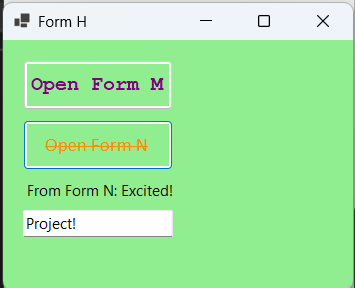


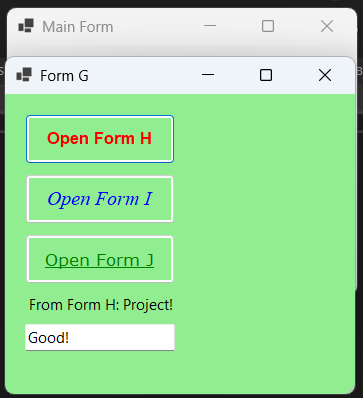


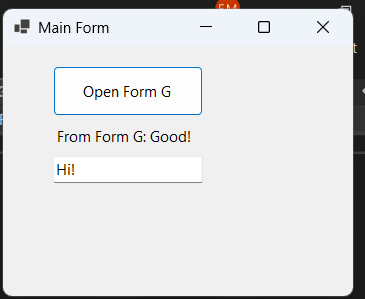












**Алгоритм побудови коду**

1. **Створення компонентів для обміну даними**:
   * Додати до кожної форми label для відображення інформації з форми-виклику.
   * Додати textbox, у який користувач може ввести текст для передачі інформації назад до форми-виклику.
2. **Обробка виклику форми**:
   * При виклику нової форми передати назву викликаючої форми та її інформацію через властивості CallingFormName і CallingFormInfo.
   * У методі OnLoad форми встановити текст label для відображення інформації про викликаючу форму.
3. **Повернення управління з форми**:
   * Після закриття форми в OnFormClosing зберегти дані з textbox у властивість ReturnedInfo.
   * Після завершення виклику відкрити label у формі-виклику та відобразити інформацію з ReturnedInfo.
4. **Передача даних між формами**:
   * Кожна форма передає свою інформацію на наступну форму через властивості, дозволяючи зберігати контекст обміну даними між формами.

**Функціональність**

* **Обмін даними між формами**: Кожна форма передає інформацію про себе наступній формі, а при поверненні відображає дані, що були введені у textbox викликаючої форми.
* **Візуалізація контексту**: За допомогою label відображається інформація про викликаючу форму, що допомагає користувачу орієнтуватися у взаємодії між формами.
* **Модальний режим**: Виклик кожної форми відбувається у модальному режимі, що забезпечує послідовну взаємодію з кожною формою.

**Висновок**

Завдання виконано, багатовіконний застосунок модифіковано для обміну даними між формами. Алгоритм передбачає передачу інформації у двох напрямах, що дозволяє формам зберігати і відображати відповідний контекст, створюючи зручнішу систему навігації між формами. Це рішення є гнучким та легким для масштабування, оскільки можна додавати нові форми, зберігаючи аналогічний спосіб обміну даними.

**Завдання 3**

За завданням власного варіанту

|  |  |
| --- | --- |
| № варіанту | Задача |
| 1 | 2 |
| 12. | Дани сторони трикутника, розрахувати його периметр та  площу |

Створити чотири варіанти застосунку (a6o в окремих проектах, a6o в одному проекті на окремих формах, які викликаються з якоїсь головной форми за допомогою кнопок чи команд меню) для вирішення задачі:

* вхідні дані i результати розрахунків відображаються на одній формі;
* вхідні дані отримуються з окремої форми;
* результати розрахунків виводяться в окрему форму;
* вхідні дані та результати розрахунків відображаються на окремих формах.

Зауваження: під час розрахунків мас перевірятися коректність даних, i разі некоректного введення мае видаватися відповідне сповіщення користувачу.

**Текст програми:**

**1. Вхідні дані i результати розрахунків відображаються на одній формі.**

***MainForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_11\_Task\_3

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

// Обробник події натискання кнопки "Розрахувати"

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

{

double hypotenuse;

// Спроба перетворити текст з TextBox у число

if (!double.TryParse(textBoxHypotenuse.Text, out hypotenuse) || hypotenuse <= 0)

{

// Виведення повідомлення про помилку при некоректному введенні

MessageBox.Show("Будь ласка, введіть коректне додатне число для гіпотенузи.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Розрахунок катета

double leg = hypotenuse / Math.Sqrt(2);

// Розрахунок периметра

double perimeter = hypotenuse + 2 \* leg;

// Розрахунок площі

double area = (leg \* leg) / 2;

// Відображення результатів

labelPerimeter.Text = $"Периметр: {perimeter:F2}";

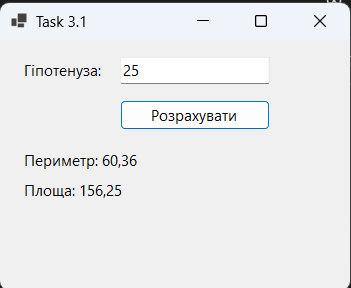
labelArea.Text = $"Площа: {area:F2}";

}

}

}

**Результат роботи програми:**



**2. Вхідні дані отримуються з окремої форми.**

***FormMain.cs:***

using System;

using System.Windows.Forms;

using Task\_3\_2;

namespace Task\_3\_2

{

public partial class FormMain : Form

{

public FormMain()

{

InitializeComponent();

}

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

{

FormInput inputForm = new FormInput();

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

{

double hypotenuse = inputForm.Hypotenuse;

CalculateAndDisplayResults(hypotenuse);

}

}

private void CalculateAndDisplayResults(double hypotenuse)

{

double leg = hypotenuse / Math.Sqrt(2);

double perimeter = hypotenuse + 2 \* leg;

double area = (leg \* leg) / 2;

labelPerimeter.Text = $"Периметр: {perimeter:F2}";

labelArea.Text = $"Площа: {area:F2}";

}

}

}

***FormInput.cs:***

using System;

using System.Windows.Forms;

namespace Task\_3\_2

{

public partial class FormInput : Form

{

public double Hypotenuse { get; private set; }

public FormInput()

{

InitializeComponent();

}

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

{

double hypotenuse;

if (!double.TryParse(textBoxHypotenuse.Text, out hypotenuse) || hypotenuse <= 0)

{

MessageBox.Show("Будь ласка, введіть коректне додатне число для гіпотенузи.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

Hypotenuse = hypotenuse;

this.DialogResult = DialogResult.OK;

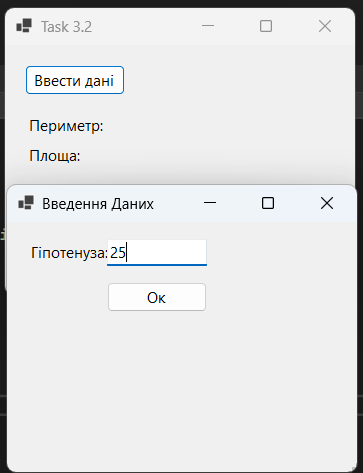
this.Close();

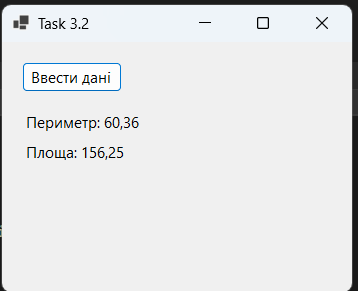
}

}

}

**Результат роботи програми:**





**3. Результати розрахунків виводяться в окрему форму.**

***FormMain.cs:***

using System;

using System.Windows.Forms;

using Task\_3\_3;

namespace Task\_3\_3

{

public partial class FormMain : Form

{

public FormMain()

{

InitializeComponent();

}

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

{

double hypotenuse;

if (!double.TryParse(textBoxHypotenuse.Text, out hypotenuse) || hypotenuse <= 0)

{

MessageBox.Show("Будь ласка, введіть коректне додатне число для гіпотенузи.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

double leg = hypotenuse / Math.Sqrt(2);

double perimeter = hypotenuse + 2 \* leg;

double area = (leg \* leg) / 2;

FormResult resultForm = new FormResult(perimeter, area);

resultForm.Show();

}

}

}

***FormResult.cs:***

using System.Windows.Forms;

namespace Task\_3\_3

{

public partial class FormResult : Form

{

public FormResult(double perimeter, double area)

{

InitializeComponent();

labelPerimeter.Text = $"Периметр: {perimeter:F2}";

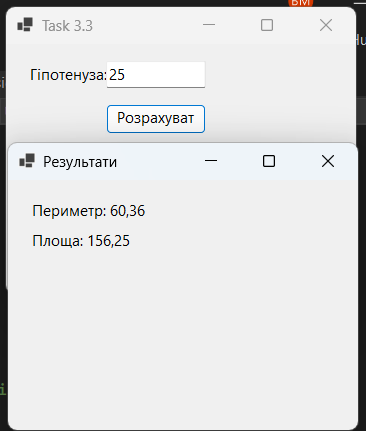
labelArea.Text = $"Площа: {area:F2}";

}

}

}

**Результат роботи програми:**



**4. Вхідні дані та результати розрахунків відображаються на окремих формах.**

***FormMain.cs:***

using System;

using System.Windows.Forms;

using Task\_3\_4;

namespace Task\_3\_4

{

public partial class FormMain : Form

{

public FormMain()

{

InitializeComponent();

}

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

{

FormInput inputForm = new FormInput();

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

{

double hypotenuse = inputForm.Hypotenuse;

double leg = hypotenuse / Math.Sqrt(2);

double perimeter = hypotenuse + 2 \* leg;

double area = (leg \* leg) / 2;

FormResult resultForm = new FormResult(perimeter, area);

resultForm.Show();

}

}

}

}

***FormResult.cs:***

using System.Windows.Forms;

namespace Task\_3\_4

{

public partial class FormResult : Form

{

public FormResult(double perimeter, double area)

{

InitializeComponent();

labelPerimeter.Text = $"Периметр: {perimeter:F2}";

labelArea.Text = $"Площа: {area:F2}";

}

}

}

***FormInput.cs:***

using System;

using System.Windows.Forms;

namespace Task\_3\_4

{

public partial class FormInput : Form

{

public double Hypotenuse { get; private set; }

public FormInput()

{

InitializeComponent();

}

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

{

double hypotenuse;

if (!double.TryParse(textBoxHypotenuse.Text, out hypotenuse) || hypotenuse <= 0)

{

MessageBox.Show("Будь ласка, введіть коректне додатне число для гіпотенузи.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

Hypotenuse = hypotenuse;

this.DialogResult = DialogResult.OK;

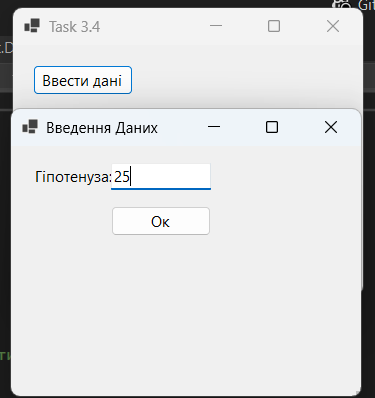
this.Close();

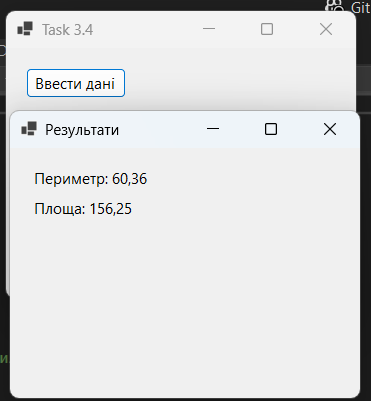
}

}

}

**Результат роботи програми:**





**Алгоритм побудови коду**

1. **Перевірка введених даних**:
   * У кожному варіанті застосунку виконується перевірка коректності введених даних для гіпотенузи (має бути числовим значенням більше 0).
   * Якщо дані некоректні, відображається повідомлення про помилку.
2. **Розрахунки**:
   * Для кожного варіанту реалізовано обчислення катета (гiпотенуза / √2), периметра (гiпотенуза + 2 \* катет) та площі (катет^2 / 2) трикутника.
3. **Реалізація чотирьох варіантів**:
   * **Варіант 1**: Вхідні дані та результати відображаються на одній формі (MainForm).
   * **Варіант 2**: Вхідні дані вводяться в окремій формі (FormInput), а результати обчислень відображаються на головній формі (FormMain).
   * **Варіант 3**: Вхідні дані вводяться на головній формі, а результати відображаються на окремій формі (FormResult).
   * **Варіант 4**: Вхідні дані вводяться на формі введення (FormInput), а результати відображаються на формі результатів (FormResult), з викликами обох форм із головної форми (FormMain).
4. **Відображення результатів**:
   * У кожному варіанті, після перевірки коректності та виконання розрахунків, результати виводяться відповідно до вимог (на головній формі чи на окремій формі).

**Функціональність**

* **Введення та перевірка даних**: У кожному варіанті передбачена можливість введення довжини гіпотенузи з валідацією даних, що захищає від некоректного введення.
* **Обчислення геометричних параметрів**: Програма обчислює параметри трикутника (периметр і площу) для заданої гіпотенузи.
* **Гнучкість інтерфейсу**: Користувач може використовувати різні форми для введення даних і перегляду результатів, що дає можливість вибору відповідного варіанту застосунку.

**Висновок**

Завдання виконано успішно, створено чотири варіанти застосунку для обчислення геометричних параметрів трикутника. Різні варіанти відображення вхідних даних та результатів покращують зручність застосунку, роблячи його більш універсальним. Алгоритм легко масштабується, дозволяючи додавати нові форми та розширювати функціональність, зберігаючи можливість вибору користувачем способу роботи.

**Завдання 4**

За завданням власного варіанту

|  |  |
| --- | --- |
| № варіанту | Вираз |
| 1 | 2 |
| 12. |  |

Створити чотири варіанти застосунку (a6o в окремих проектах, a6o в одному проекті на окремих формах, які викликаються з якоїсь головной форми за допомогою кнопок чи команд меню) для розрахунку виразу:

* вхідні дані i результати розрахунків виразу відображаються на одній формі;
* вхідні дані отримуються з окремої форми;
* результати розрахунків виводяться в окрему форму;
* вхідні дані та результати розрахунків відображаються на окремих формах.

Зауваження: під час розрахунків мас перевірятися коректність даних, i разі некоректного введення має видаватися відповідне сповіщення користувачу.

**Текст програми:**

**1. Вхідні дані i результати розрахунків виразу відображаються на одній формі.**

***MainForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_11\_Task\_4

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

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

{

int x;

double y, z;

// Перевірка коректності введення x

if (!int.TryParse(txtX.Text, out x) || x <= 0)

{

MessageBox.Show("Введіть коректне додатне ціле число для x.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення y

if (!double.TryParse(txtY.Text, out y))

{

MessageBox.Show("Введіть коректне число для y.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення z

if (!double.TryParse(txtZ.Text, out z))

{

MessageBox.Show("Введіть коректне число для z.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

double result = CalculateExpression(x, y, z);

lblResult.Text = $"Результат: {result}";

}

catch (DivideByZeroException ex)

{

MessageBox.Show(ex.Message, "Помилка обчислення", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show("Сталася помилка під час обчислення: " + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private double CalculateExpression(int x, double y, double z)

{

double sum = 0.0;

for (int i = 1; i <= x; i++)

{

double product = 1.0;

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

{

double denominator = i + j + z;

if (denominator == 0)

{

throw new DivideByZeroException("Ділення на нуль при i + j + z.");

}

product \*= (x \* y \* z) / denominator;

}

sum += product;

}

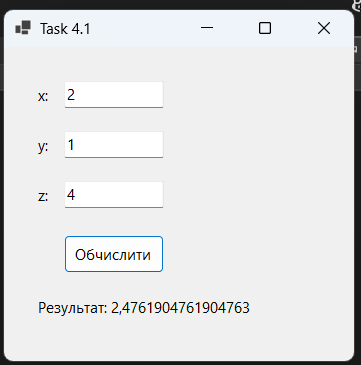
return sum;

}

}

}

**Результат роботи програми:**



**2. Вхідні дані отримуються з окремої форми.**

***MainForm.cs:***

using System;

using System.Windows.Forms;

namespace Task\_4\_2

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

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

{

// Створюємо екземпляр форми введення

InputForm inputForm = new InputForm();

// Відображаємо форму введення як модальну

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

{

// Отримуємо введені значення

int x = inputForm.X;

double y = inputForm.Y;

double z = inputForm.Z;

try

{

double result = CalculateExpression(x, y, z);

lblResult.Text = $"Результат: {result}";

}

catch (DivideByZeroException ex)

{

MessageBox.Show(ex.Message, "Помилка обчислення", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show("Сталася помилка під час обчислення: " + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private double CalculateExpression(int x, double y, double z)

{

double sum = 0.0;

for (int i = 1; i <= x; i++)

{

double product = 1.0;

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

{

double denominator = i + j + z;

if (denominator == 0)

{

throw new DivideByZeroException("Ділення на нуль при i + j + z.");

}

product \*= (x \* y \* z) / denominator;

}

sum += product;

}

return sum;

}

}

}

***InputForm.cs:***

using System;

using System.Windows.Forms;

namespace Task\_4\_2

{

public partial class InputForm : Form

{

public int X { get; private set; }

public double Y { get; private set; }

public double Z { get; private set; }

public InputForm()

{

InitializeComponent();

}

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

{

int x;

double y, z;

// Перевірка коректності введення x

if (!int.TryParse(txtX.Text, out x) || x <= 0)

{

MessageBox.Show("Введіть коректне додатне ціле число для x.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення y

if (!double.TryParse(txtY.Text, out y))

{

MessageBox.Show("Введіть коректне число для y.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення z

if (!double.TryParse(txtZ.Text, out z))

{

MessageBox.Show("Введіть коректне число для z.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Збереження введених значень

X = x;

Y = y;

Z = z;

// Закриваємо форму з результатом DialogResult.OK

this.DialogResult = DialogResult.OK;

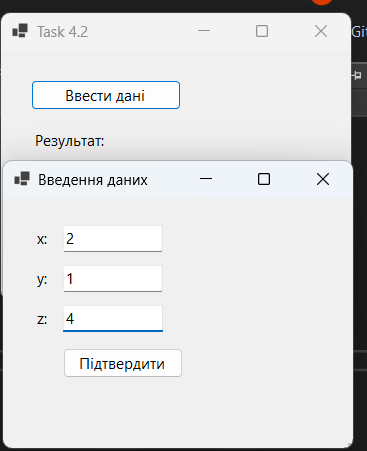
this.Close();

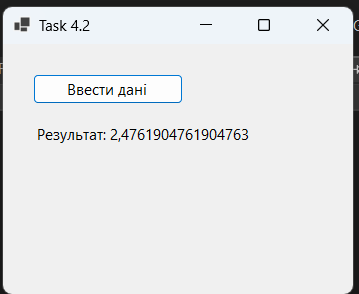
}

}

}

**Результат роботи програми:**





**3. Результати розрахунків виводяться в окрему форму.**

***MainForm.cs:***

using System;

using System.Windows.Forms;

namespace Task\_4\_3

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

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

{

int x;

double y, z;

// Перевірка коректності введення x

if (!int.TryParse(txtX.Text, out x) || x <= 0)

{

MessageBox.Show("Введіть коректне додатне ціле число для x.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення y

if (!double.TryParse(txtY.Text, out y))

{

MessageBox.Show("Введіть коректне число для y.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення z

if (!double.TryParse(txtZ.Text, out z))

{

MessageBox.Show("Введіть коректне число для z.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

double result = CalculateExpression(x, y, z);

// Відкриваємо форму результатів і передаємо результат

ResultsForm resultsForm = new ResultsForm(result);

resultsForm.Show();

}

catch (DivideByZeroException ex)

{

MessageBox.Show(ex.Message, "Помилка обчислення", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show("Сталася помилка під час обчислення: " + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private double CalculateExpression(int x, double y, double z)

{

double sum = 0.0;

for (int i = 1; i <= x; i++)

{

double product = 1.0;

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

{

double denominator = i + j + z;

if (denominator == 0)

{

throw new DivideByZeroException("Ділення на нуль при i + j + z.");

}

product \*= (x \* y \* z) / denominator;

}

sum += product;

}

return sum;

}

}

}

***ResultsForm.cs:***

using System;

using System.Windows.Forms;

namespace Task\_4\_3

{

public partial class ResultsForm : Form

{

public ResultsForm(double result)

{

InitializeComponent();

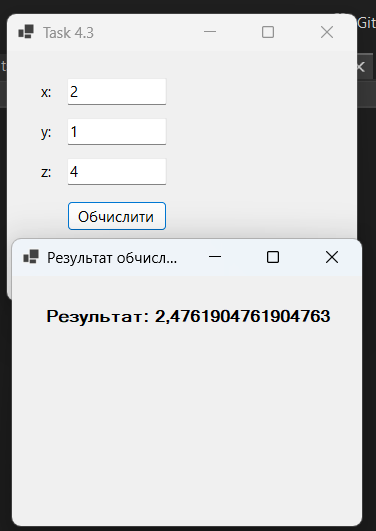
lblResult.Text = $"Результат: {result}";

}

}

}

**Результат роботи програми:**



**4. Вхідні дані та результати розрахунків відображаються на окремих формах.**

***MainForm.cs:***

using System;

using System.Windows.Forms;

namespace Task\_4\_4

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

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

{

// Відкриваємо форму введення даних

InputForm inputForm = new InputForm();

inputForm.Show();

}

}

}

***InputForm.cs:***

using System;

using System.Windows.Forms;

namespace Task\_4\_4

{

public partial class InputForm : Form

{

public InputForm()

{

InitializeComponent();

}

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

{

int x;

double y, z;

// Перевірка коректності введення x

if (!int.TryParse(txtX.Text, out x) || x <= 0)

{

MessageBox.Show("Введіть коректне додатне ціле число для x.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення y

if (!double.TryParse(txtY.Text, out y))

{

MessageBox.Show("Введіть коректне число для y.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Перевірка коректності введення z

if (!double.TryParse(txtZ.Text, out z))

{

MessageBox.Show("Введіть коректне число для z.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

double result = CalculateExpression(x, y, z);

// Відкриваємо форму результатів і передаємо результат

ResultsForm resultsForm = new ResultsForm(result);

resultsForm.Show();

// Закриваємо форму введення даних

this.Close();

}

catch (DivideByZeroException ex)

{

MessageBox.Show(ex.Message, "Помилка обчислення", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show("Сталася помилка під час обчислення: " + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private double CalculateExpression(int x, double y, double z)

{

double sum = 0.0;

for (int i = 1; i <= x; i++)

{

double product = 1.0;

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

{

double denominator = i + j + z;

if (denominator == 0)

{

throw new DivideByZeroException("Ділення на нуль при i + j + z.");

}

product \*= (x \* y \* z) / denominator;

}

sum += product;

}

return sum;

}

}

}

***ResultsForm.cs:***

using System;

using System.Windows.Forms;

namespace Task\_4\_4

{

public partial class ResultsForm : Form

{

public ResultsForm(double result)

{

InitializeComponent();

lblResult.Text = $"Результат: {result}";

}

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

{

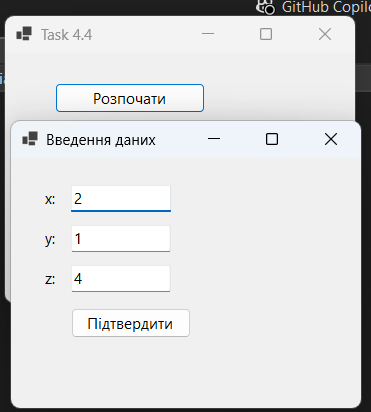
this.Close();

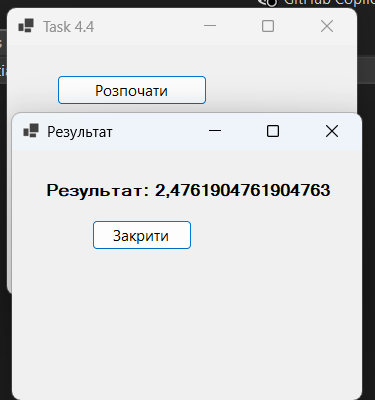
}

}

}

**Результат роботи програми:**





**Алгоритм побудови коду**

1. **Перевірка коректності введених даних**:
   * Для кожного варіанту реалізована перевірка коректності введення числових значень змінних x, y, z.
   * Якщо дані некоректні, відображається повідомлення про помилку, що запобігає помилковим обчисленням.
2. **Обчислення виразу**:
   * Реалізовано метод CalculateExpression, який обчислює заданий вираз, використовуючи вкладені цикли.
   * У разі ділення на нуль генерується виключення DivideByZeroException, яке обробляється та відображається як повідомлення користувачу.
3. **Реалізація чотирьох варіантів**:
   * **Варіант 1**: Вхідні дані та результати розміщені на одній формі. Після введення значень і натискання кнопки Розрахувати відображаються результати.
   * **Варіант 2**: Вхідні дані вводяться на окремій формі (InputForm), після чого обчислення проводяться на головній формі, де й відображаються результати.
   * **Варіант 3**: На головній формі вводяться дані, а результати виводяться на окрему форму результатів (ResultsForm).
   * **Варіант 4**: Вхідні дані вводяться на формі InputForm, а результати відображаються на іншій формі ResultsForm.
4. **Відображення результатів**:
   * У кожному варіанті після успішного обчислення результат виводиться на екран у відповідній формі.

**Функціональність**

* **Валідація введення**: Кожен варіант забезпечує перевірку введених даних, запобігаючи некоректним обчисленням.
* **Гнучкість інтерфейсу**: Користувач має можливість вибору варіанту введення та виведення даних, що дозволяє більш ефективно виконувати обчислення відповідно до потреб.
* **Модульність обчислення**: Реалізація обчислювального методу як окремої функції дозволяє легко використовувати його в різних варіантах форми.

**Висновок**

Завдання було успішно виконане, створено чотири варіанти застосунку для обчислення виразу. Кожен варіант забезпечує різний спосіб введення та виведення даних, що робить застосунок більш гнучким і зручним для користувача. Архітектура програми дозволяє легко масштабувати застосунок та додавати нові форми чи функції, зберігаючи логіку обчислень окремо.

**Завдання 5**

Створити застосунок, який на головній формі дозволяв би вибирати місто країни згідно до свого варіанту та відобразити фото та короткий опис на додатковій формі

|  |  |  |
| --- | --- | --- |
| **№ варіанту** | **Країна** | **Кількість міст** |
| 12. | Єгипет | 5 |

**Текст програми:**

***MainForm.cs:***

using System;

using System.Collections.Generic;

using System.Windows.Forms;

using System.Drawing;

namespace Lab\_11\_Task\_5

{

public partial class MainForm : Form

{

// Список міст

private List<City> cities;

public MainForm()

{

InitializeComponent();

InitializeCities();

PopulateComboBox();

}

private void InitializeCities()

{

// Перетворюємо масиви байтів у зображення

Image cairoImage = ByteArrayToImage(Properties.Resources.cairo);

Image alexandriaImage = ByteArrayToImage(Properties.Resources.alexandria);

Image luxorImage = ByteArrayToImage(Properties.Resources.luxor);

Image hurghadaImage = ByteArrayToImage(Properties.Resources.hurghada);

Image sharmImage = ByteArrayToImage(Properties.Resources.sharm);

// Створюємо об'єкти міст

cities = new List<City>

{

new City("Каїр", "Столиця Єгипту, найбільше місто Африки.", cairoImage),

new City("Александрія", "Відоме портове місто на Середземному морі.", alexandriaImage),

new City("Луксор", "Місто-музей під відкритим небом з древніми храмами.", luxorImage),

new City("Хургада", "Популярний курорт на Червоному морі.", hurghadaImage),

new City("Шарм-ель-Шейх", "Відомий курорт з чудовими місцями для дайвінгу.", sharmImage)

};

}

// Метод для перетворення масиву байтів у зображення

private Image ByteArrayToImage(byte[] byteArray)

{

using (MemoryStream ms = new MemoryStream(byteArray))

{

return Image.FromStream(ms);

}

}

private void PopulateComboBox()

{

foreach (var city in cities)

{

comboBoxCities.Items.Add(city.Name);

}

}

private void comboBoxCities\_SelectedIndexChanged(object sender, EventArgs e)

{

int selectedIndex = comboBoxCities.SelectedIndex;

if (selectedIndex >= 0)

{

City selectedCity = cities[selectedIndex];

InfoForm form2 = new InfoForm(selectedCity);

form2.ShowDialog();

}

}

}

}

***InfoForm.cs:***

using System;

using System.Windows.Forms;

using System.Drawing;

namespace Lab\_11\_Task\_5

{

public partial class InfoForm : Form

{

private City city;

public InfoForm(City selectedCity)

{

InitializeComponent();

city = selectedCity;

DisplayCityInfo();

}

private void DisplayCityInfo()

{

labelCityName.Text = city.Name;

pictureBoxCity.Image = city.Image;

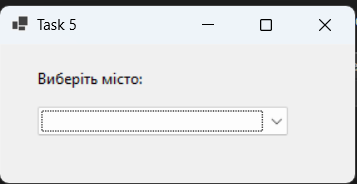
textBoxDescription.Text = city.Description;

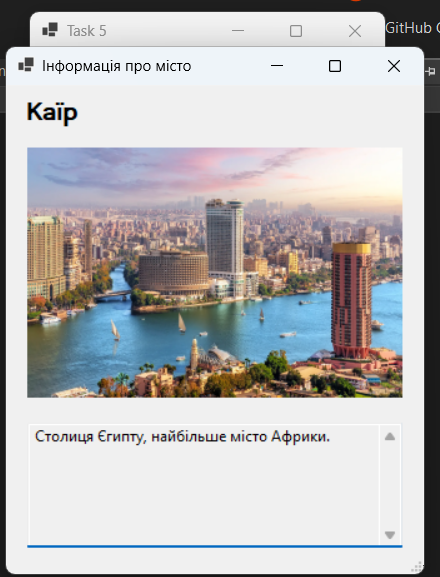
}

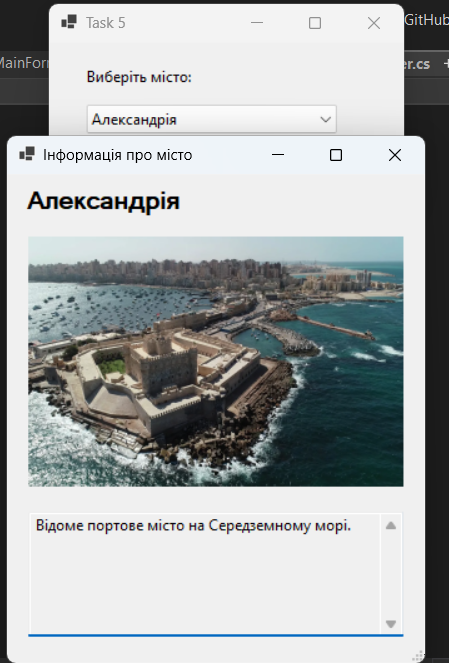
}

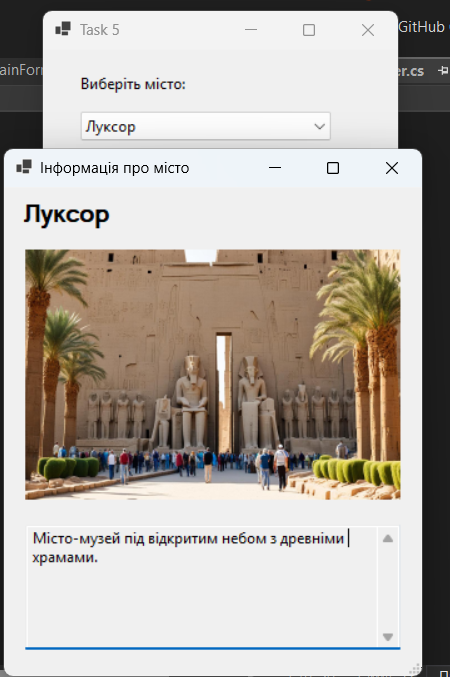
}

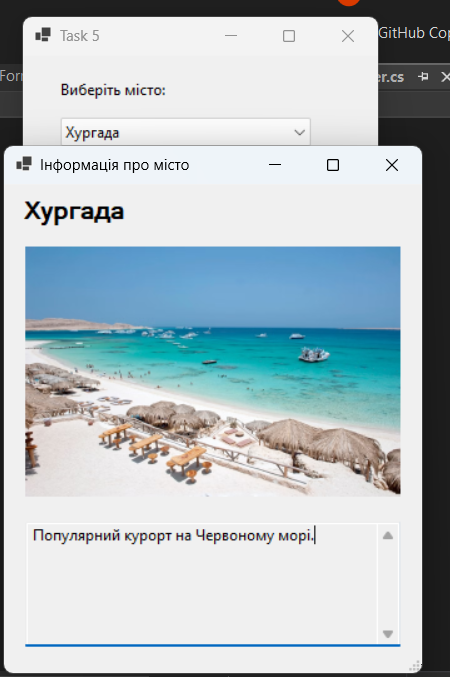
**Результат роботи програми:**

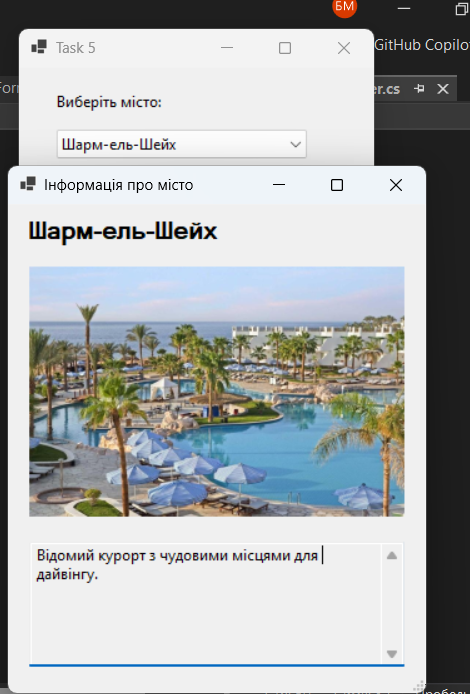












**Алгоритм побудови коду**

1. **Ініціалізація даних про міста**:
   * Створити список об'єктів типу City, кожен з яких містить назву, опис і зображення міста.
   * Зображення зберігаються в ресурсах програми й конвертуються з масиву байтів у формат Image.
2. **Відображення списку міст у ComboBox**:
   * Додати назви міст до випадаючого списку (comboBoxCities) на головній формі.
3. **Обробка вибору міста**:
   * При виборі міста у comboBoxCities визначається індекс вибраного елемента.
   * На основі цього індексу створюється об'єкт форми InfoForm, якому передається інформація про вибране місто, і форма відкривається у модальному режимі.
4. **Відображення інформації на додатковій формі**:
   * У формі InfoForm місто передається як параметр до конструктора, де викликається метод DisplayCityInfo.
   * Метод DisplayCityInfo заповнює елементи форми інформацією про місто: назва відображається в labelCityName, зображення в pictureBoxCity, а опис у textBoxDescription.

**Функціональність**

* **Вибір міста з випадаючого списку**: Користувач може вибрати будь-яке з п’яти міст Єгипту.
* **Відображення детальної інформації**: Після вибору міста відкривається нова форма з назвою, зображенням та коротким описом міста.
* **Дружній інтерфейс**: Застосунок забезпечує інтуїтивно зрозумілий інтерфейс із легким доступом до інформації про кожне місто.

**Висновок**

Завдання було успішно виконане. Створено застосунок, який дозволяє переглядати інформацію про п'ять міст Єгипту, відображаючи їхні зображення та короткі описи на додатковій формі. Архітектура застосунку забезпечує гнучкість у додаванні нових міст, що робить його легко масштабованим.

**Завдання 6**

До форми застосунку із четвертого завдання четвертой роботи додати кнопку із назвою, наприклад, Конфігурація a6o Параметри, яка 6 викликала окрему форму для налаштування кольорово-розмірні параметри ()не менше п'яти) компонентів основной форми. Для налаштування рекомендується використовувати компоненти діалогів: ColorDialog та FontDialog.

**Текст програми:**

***MainForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_11\_Task\_6

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

private void ButtonSubmit\_Click(object sender, EventArgs e) // Обробник натискання кнопки Submit

{

// Перевірка на заповненість полів

if (string.IsNullOrWhiteSpace(textBoxName.Text) ||

string.IsNullOrWhiteSpace(textBoxEmail.Text) ||

string.IsNullOrWhiteSpace(textBoxPhone.Text) ||

comboBoxAge.SelectedIndex == -1 ||

comboBoxCuisine.SelectedIndex == -1 ||

(!radioButtonReason1.Checked && !radioButtonReason2.Checked) ||

(!radioButtonYes.Checked && !radioButtonNo.Checked))

{

MessageBox.Show("Будь ласка, заповніть усі обов'язкові поля.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return; // Завершення

}

// Збір даних

var message = string.Join("\n", new[]

{

$"Ім'я: {textBoxName.Text}",

$"Пошта: {textBoxEmail.Text}",

$"Телефон: {textBoxPhone.Text}",

$"Дата відвідування: {dateTimePickerVisit.Value.ToShortDateString()}",

$"Вік: {comboBoxAge.SelectedItem}",

$"Улюблена кухня: {comboBoxCuisine.SelectedItem}",

$"Страви: {textBoxDishes.Text}",

$"Чому обрали наш ресторан: {(radioButtonReason1.Checked ? radioButtonReason1.Text : radioButtonReason2.Text)}",

$"Чи порекомендуєте ресторан: {(radioButtonYes.Checked ? "Так" : "Ні")}"

});

// Виведення повідомлення

MessageBox.Show(message, "Інформація про відвідувача");

}

private void ButtonClear\_Click(object sender, EventArgs e) // Обробник натискання кнопки Clear

{

// Очищення полів

textBoxName.Clear();

textBoxEmail.Clear();

textBoxPhone.Clear();

dateTimePickerVisit.Value = DateTime.Now;

comboBoxAge.SelectedIndex = -1;

comboBoxCuisine.SelectedIndex = -1;

textBoxDishes.Clear();

// Очищення радіо-кнопок

radioButtonReason1.Checked = false;

radioButtonReason2.Checked = false;

radioButtonYes.Checked = false;

radioButtonNo.Checked = false;

}

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

{

// Відкриття форми налаштувань

ConfigForm configForm = new ConfigForm(this);

configForm.Show();

}

}

}

***ConfigForm.cs:***

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab\_11\_Task\_6

{

public partial class ConfigForm : Form

{

private MainForm mainForm;

public ConfigForm(MainForm form)

{

InitializeComponent();

mainForm = form;

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.labelTitle.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.labelTitle.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxContactInfo.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxContactInfo.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxPersonalInfo.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxPersonalInfo.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxEvaluation.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxEvaluation.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.buttonSubmit.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.buttonSubmit.Font = fontDialog1.Font;

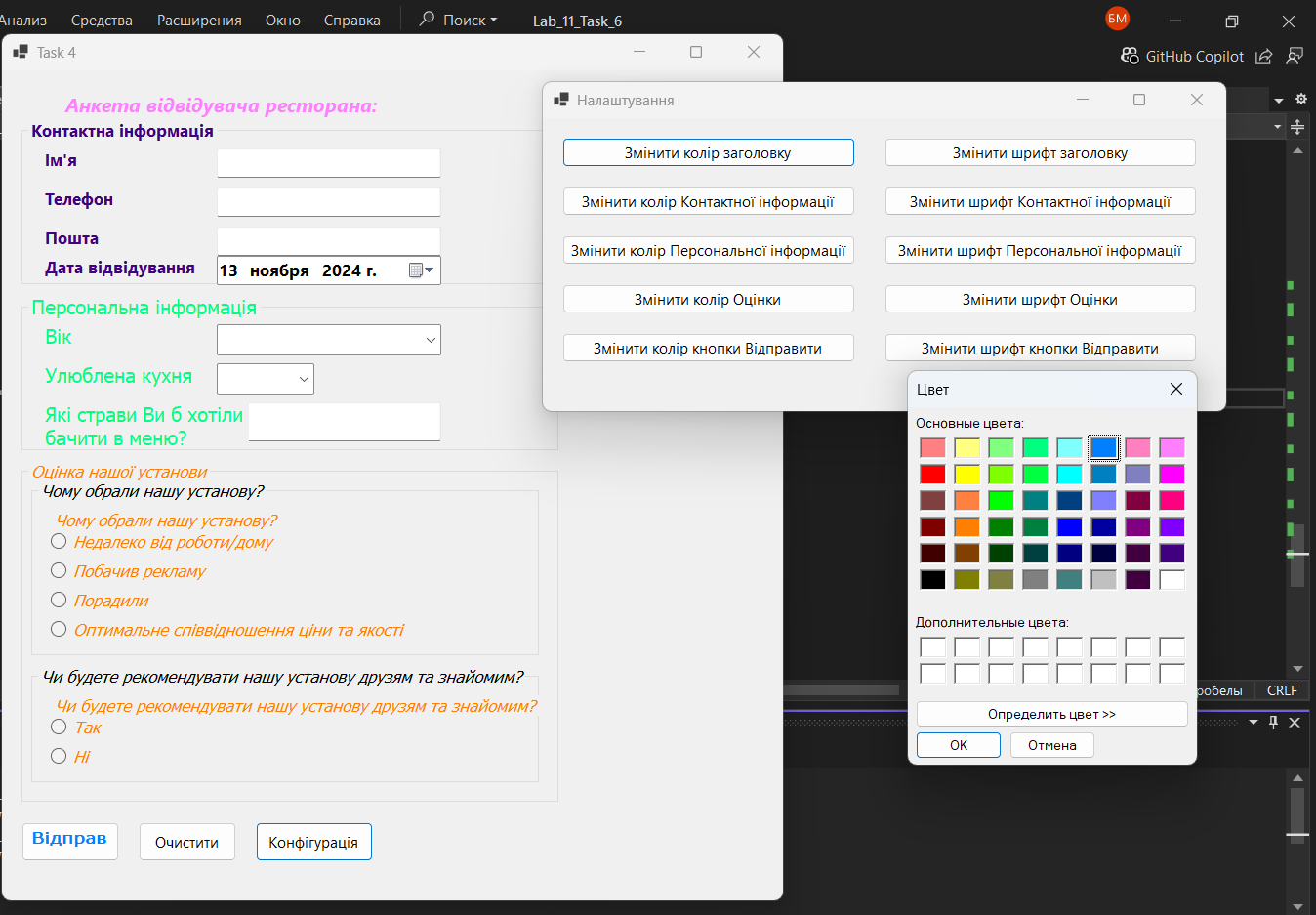
}

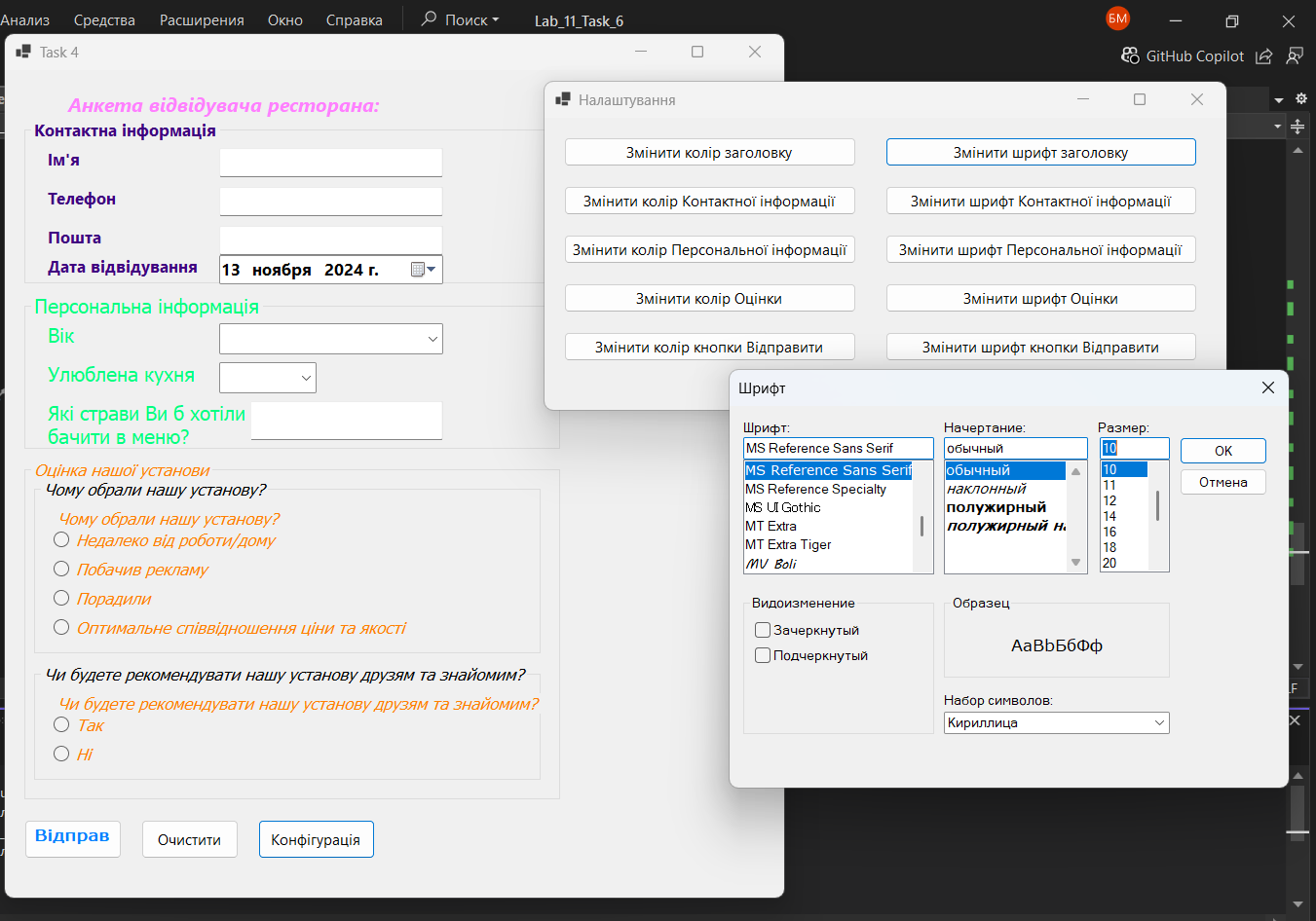
}

}

}

**Результат роботи програми:**





**Алгоритм побудови коду**

1. **Додавання кнопки конфігурації**:
   * На головній формі (MainForm) додати кнопку (наприклад, з назвою "Конфігурація").
   * Додати обробник події для кнопки, який відкриває форму конфігурації (ConfigForm), передаючи посилання на головну форму для доступу до її компонентів.
2. **Форма конфігурації (ConfigForm)**:
   * Форма отримує доступ до головної форми через переданий параметр MainForm у конструкторі.
   * Додає кнопки для налаштування кожного компонента, що дозволяють змінювати його колір і шрифт.
   * Кожна кнопка викликає відповідний діалог (ColorDialog або FontDialog) для вибору кольору або шрифту, а результат застосовується до компонентів головної форми.
3. **Налаштування компонентів головної форми**:
   * Обробники подій для кнопок у ConfigForm оновлюють параметри (колір тексту, шрифт) вибраного компонента головної форми за результатами вибору в діалогових вікнах.

**Функціональність**

* **Конфігурація стилю компонентів**: Форма ConfigForm дозволяє налаштовувати параметри кольору і шрифту для різних компонентів головної форми, таких як заголовок (labelTitle), групи полів (groupBoxContactInfo, groupBoxPersonalInfo, groupBoxEvaluation) та кнопка Submit.
* **Інтерактивний вибір стилю**: Завдяки використанню ColorDialog та FontDialog користувач може легко вибрати будь-який доступний колір або шрифт, що робить застосунок більш налаштовуваним.

**Висновок**

Завдання виконано успішно, додано можливість налаштовувати вигляд головної форми. Нова функціональність підвищує гнучкість застосунку, дозволяючи користувачу змінювати стилі ключових компонентів. Завдяки реалізації через окрему форму конфігурації (ConfigForm) код залишається структурованим і легко масштабованим для додавання нових параметрів налаштування.

**Завдання 7\***

Модифікувати застосунок минулого завдання додавши до конфігураційної форми можливість зберігати та завантажувати конфігураційні параметри компонентів головной форми у(з) зовнішніх (за бажанням текстових a6o двійкових) файлах, застосовуючи для цього компоненти OpenFileDialog та SaveFileDialog. Передбачити можливість автоматичного завантаження параметрів компонентів головной форми під час її запуску.

**Текст програми:**

***MainForm.cs:***

using System;

using Newtonsoft.Json;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace Lab\_11\_Task\_7

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

string configFile = Path.Combine(Application.StartupPath, "default.cfg");

LoadConfiguration(configFile);

}

private void ButtonSubmit\_Click(object sender, EventArgs e) // Обробник натискання кнопки Submit

{

// Перевірка на заповненість полів

if (string.IsNullOrWhiteSpace(textBoxName.Text) ||

string.IsNullOrWhiteSpace(textBoxEmail.Text) ||

string.IsNullOrWhiteSpace(textBoxPhone.Text) ||

comboBoxAge.SelectedIndex == -1 ||

comboBoxCuisine.SelectedIndex == -1 ||

(!radioButtonReason1.Checked && !radioButtonReason2.Checked) ||

(!radioButtonYes.Checked && !radioButtonNo.Checked))

{

MessageBox.Show("Будь ласка, заповніть усі обов'язкові поля.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return; // Завершення

}

// Збір даних

string name = textBoxName.Text;

string email = textBoxEmail.Text;

string phone = textBoxPhone.Text;

DateTime visitDate = dateTimePickerVisit.Value;

string age = comboBoxAge.SelectedItem?.ToString();

string cuisine = comboBoxCuisine.SelectedItem?.ToString();

string dishes = textBoxDishes.Text;

// Вибір причини та рекомендації

string reason = radioButtonReason1.Checked ? radioButtonReason1.Text : radioButtonReason2.Text;

string recommend = radioButtonYes.Checked ? "Так" : "Ні";

// Формування повідомлення

string message = $"Ім'я: {name}\nПошта: {email}\nТелефон: {phone}\nДата відвідування: {visitDate.ToShortDateString()}\n" +

$"Вік: {age}\nУлюблена кухня: {cuisine}\nСтрави: {dishes}\nЧому обрали наш ресторан: {reason}\n" +

$"Чи порекомендуєте ресторан: {recommend}";

MessageBox.Show(message, "Інформація про відвідувача"); // Виведення інформації

}

private void ButtonClear\_Click(object sender, EventArgs e) // Обробник натискання кнопки Clear

{

// Очищення полів

textBoxName.Clear();

textBoxEmail.Clear();

textBoxPhone.Clear();

dateTimePickerVisit.Value = DateTime.Now;

comboBoxAge.SelectedIndex = -1;

comboBoxCuisine.SelectedIndex = -1;

textBoxDishes.Clear();

// Очищення радіо-кнопок

radioButtonReason1.Checked = false;

radioButtonReason2.Checked = false;

radioButtonYes.Checked = false;

radioButtonNo.Checked = false;

}

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

{

// Відкриття форми налаштувань

ConfigForm configForm = new ConfigForm(this);

configForm.Show();

}

public void LoadConfiguration(string fileName)

{

try

{

if (File.Exists(fileName))

{

using (StreamReader reader = new StreamReader(fileName))

{

// Десеріалізація конфігурації

string json = reader.ReadToEnd();

Configuration config = JsonConvert.DeserializeObject<Configuration>(json);

// Застосування параметрів

ApplyConfiguration(labelTitle, config.LabelTitleForeColorArgb, config.LabelTitleFontName, config.LabelTitleFontSize, config.LabelTitleFontStyle);

ApplyConfiguration(groupBoxContactInfo, config.GroupBoxContactInfoForeColorArgb, config.GroupBoxContactInfoFontName, config.GroupBoxContactInfoFontSize, config.GroupBoxContactInfoFontStyle);

ApplyConfiguration(groupBoxPersonalInfo, config.GroupBoxPersonalInfoForeColorArgb, config.GroupBoxPersonalInfoFontName, config.GroupBoxPersonalInfoFontSize, config.GroupBoxPersonalInfoFontStyle);

ApplyConfiguration(groupBoxEvaluation, config.GroupBoxEvaluationForeColorArgb, config.GroupBoxEvaluationFontName, config.GroupBoxEvaluationFontSize, config.GroupBoxEvaluationFontStyle);

ApplyConfiguration(buttonSubmit, config.ButtonSubmitForeColorArgb, config.ButtonSubmitFontName, config.ButtonSubmitFontSize, config.ButtonSubmitFontStyle);

}

}

}

catch (Exception ex)

{

MessageBox.Show("Помилка при завантаженні конфігурації: " + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ApplyConfiguration(Control control, int foreColorArgb, string fontName, float fontSize, FontStyle fontStyle)

{

control.ForeColor = Color.FromArgb(foreColorArgb);

control.Font = new Font(fontName, fontSize, fontStyle);

}

}

}

***ConfigForm.cs:***

using Lab\_11\_Task\_7;

using System;

using Newtonsoft.Json;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab\_11\_Task\_7

{

public partial class ConfigForm : Form

{

private MainForm mainForm;

public ConfigForm(MainForm form)

{

InitializeComponent();

mainForm = form;

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.labelTitle.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.labelTitle.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxContactInfo.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxContactInfo.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxPersonalInfo.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxPersonalInfo.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxEvaluation.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.groupBoxEvaluation.Font = fontDialog1.Font;

}

}

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

{

if (colorDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.buttonSubmit.ForeColor = colorDialog1.Color;

}

}

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

{

if (fontDialog1.ShowDialog() == DialogResult.OK)

{

mainForm.buttonSubmit.Font = fontDialog1.Font;

}

}

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

{

// Створення об'єкта конфігурації

Configuration config = new Configuration();

// Збір параметрів з головної форми

SetFontAndColorConfig(config, mainForm.labelTitle, "LabelTitle");

SetFontAndColorConfig(config, mainForm.groupBoxContactInfo, "GroupBoxContactInfo");

SetFontAndColorConfig(config, mainForm.groupBoxPersonalInfo, "GroupBoxPersonalInfo");

SetFontAndColorConfig(config, mainForm.groupBoxEvaluation, "GroupBoxEvaluation");

SetFontAndColorConfig(config, mainForm.buttonSubmit, "ButtonSubmit");

// Виклик SaveFileDialog

saveFileDialog1.Filter = "Config Files|\*.cfg|All Files|\*.\*";

if (saveFileDialog1.ShowDialog() == DialogResult.OK)

{

SaveConfigToFile(saveFileDialog1.FileName, config);

}

// Збереження у файл за замовчуванням

string defaultConfigFile = Path.Combine(Application.StartupPath, "default.cfg");

SaveConfigToFile(defaultConfigFile, config);

}

private void SetFontAndColorConfig(Configuration config, Control control, string prefix)

{

var font = control.Font;

typeof(Configuration).GetProperty($"{prefix}ForeColorArgb")?.SetValue(config, control.ForeColor.ToArgb());

typeof(Configuration).GetProperty($"{prefix}FontName")?.SetValue(config, font.FontFamily.Name);

typeof(Configuration).GetProperty($"{prefix}FontSize")?.SetValue(config, font.Size);

typeof(Configuration).GetProperty($"{prefix}FontStyle")?.SetValue(config, font.Style);

}

private void SaveConfigToFile(string fileName, Configuration config)

{

try

{

using (StreamWriter writer = new StreamWriter(fileName))

{

string json = JsonConvert.SerializeObject(config);

writer.Write(json);

}

MessageBox.Show("Конфігурація успішно збережена.", "Збереження", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show("Помилка при збереженні конфігурації: " + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

openFileDialog1.Filter = "Config Files|\*.cfg|All Files|\*.\*";

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

string fileName = openFileDialog1.FileName;

try

{

using (StreamReader reader = new StreamReader(fileName))

{

string json = reader.ReadToEnd();

Configuration config = JsonConvert.DeserializeObject<Configuration>(json);

// Масив з елементами для налаштування і відповідними параметрами конфігурації

var elements = new (Control control, string fontName, float fontSize, FontStyle fontStyle, int colorArgb)[]

{

(mainForm.labelTitle, config.LabelTitleFontName, config.LabelTitleFontSize, config.LabelTitleFontStyle, config.LabelTitleForeColorArgb),

(mainForm.groupBoxContactInfo, config.GroupBoxContactInfoFontName, config.GroupBoxContactInfoFontSize, config.GroupBoxContactInfoFontStyle, config.GroupBoxContactInfoForeColorArgb),

(mainForm.groupBoxPersonalInfo, config.GroupBoxPersonalInfoFontName, config.GroupBoxPersonalInfoFontSize, config.GroupBoxPersonalInfoFontStyle, config.GroupBoxPersonalInfoForeColorArgb),

(mainForm.groupBoxEvaluation, config.GroupBoxEvaluationFontName, config.GroupBoxEvaluationFontSize, config.GroupBoxEvaluationFontStyle, config.GroupBoxEvaluationForeColorArgb),

(mainForm.buttonSubmit, config.ButtonSubmitFontName, config.ButtonSubmitFontSize, config.ButtonSubmitFontStyle, config.ButtonSubmitForeColorArgb)

};

// Застосування конфігурації до елементів форми

foreach (var (control, fontName, fontSize, fontStyle, colorArgb) in elements)

{

control.ForeColor = Color.FromArgb(colorArgb);

control.Font = new Font(fontName, fontSize, fontStyle);

}

MessageBox.Show("Конфігурація успішно завантажена.", "Завантаження", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

MessageBox.Show("Помилка при завантаженні конфігурації: " + ex.Message, "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

***Configuration.cs:***

using System;

using System.Drawing;

namespace Lab\_11\_Task\_7

{

[Serializable]

public class Configuration

{

// Параметри для labelTitle

public int LabelTitleForeColorArgb { get; set; }

public string LabelTitleFontName { get; set; }

public float LabelTitleFontSize { get; set; }

public FontStyle LabelTitleFontStyle { get; set; }

// Параметри для groupBoxContactInfo

public int GroupBoxContactInfoForeColorArgb { get; set; }

public string GroupBoxContactInfoFontName { get; set; }

public float GroupBoxContactInfoFontSize { get; set; }

public FontStyle GroupBoxContactInfoFontStyle { get; set; }

// Параметри для groupBoxPersonalInfo

public int GroupBoxPersonalInfoForeColorArgb { get; set; }

public string GroupBoxPersonalInfoFontName { get; set; }

public float GroupBoxPersonalInfoFontSize { get; set; }

public FontStyle GroupBoxPersonalInfoFontStyle { get; set; }

// Параметри для groupBoxEvaluation

public int GroupBoxEvaluationForeColorArgb { get; set; }

public string GroupBoxEvaluationFontName { get; set; }

public float GroupBoxEvaluationFontSize { get; set; }

public FontStyle GroupBoxEvaluationFontStyle { get; set; }

// Параметри для buttonSubmit

public int ButtonSubmitForeColorArgb { get; set; }

public string ButtonSubmitFontName { get; set; }

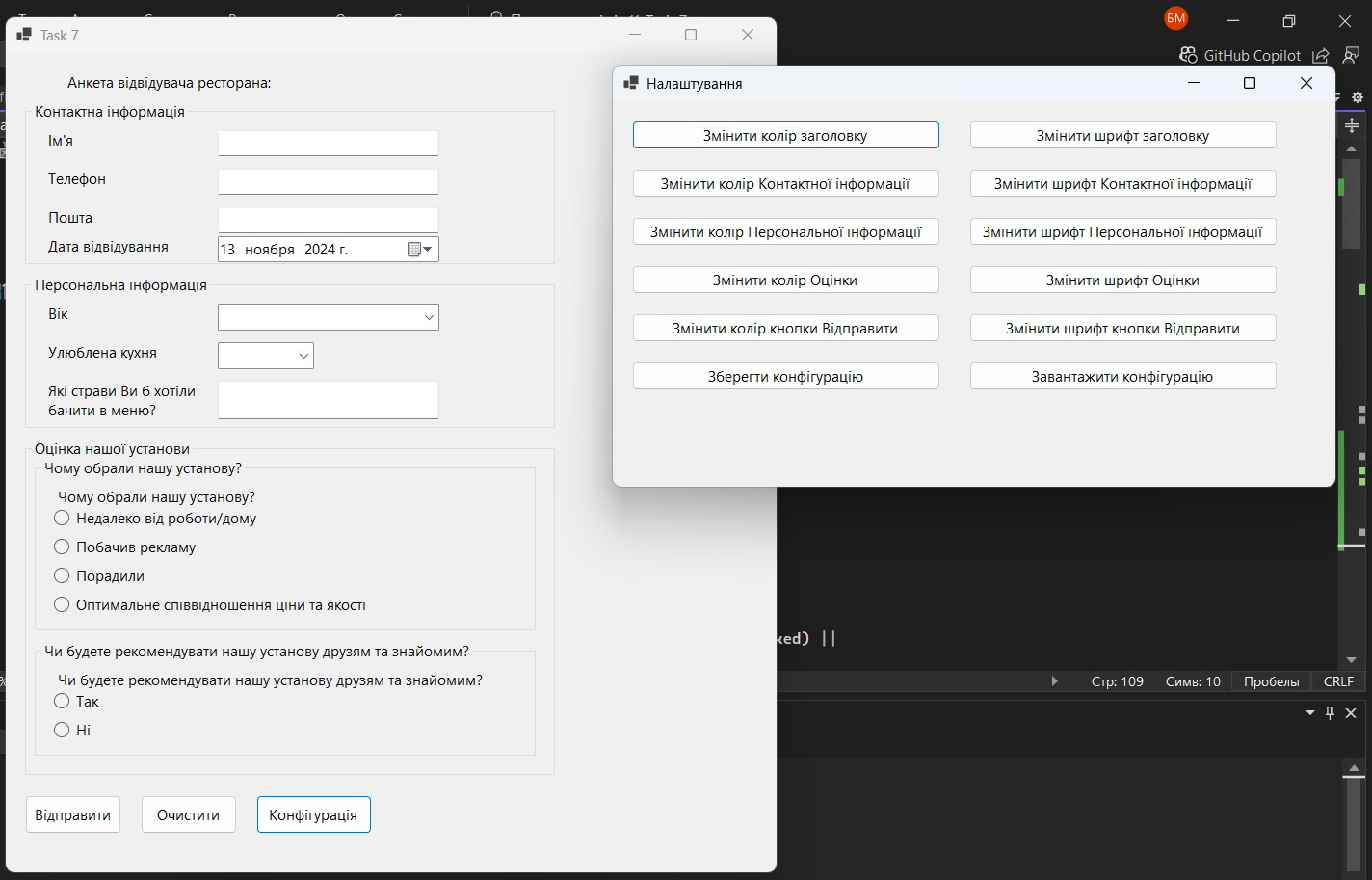
public float ButtonSubmitFontSize { get; set; }

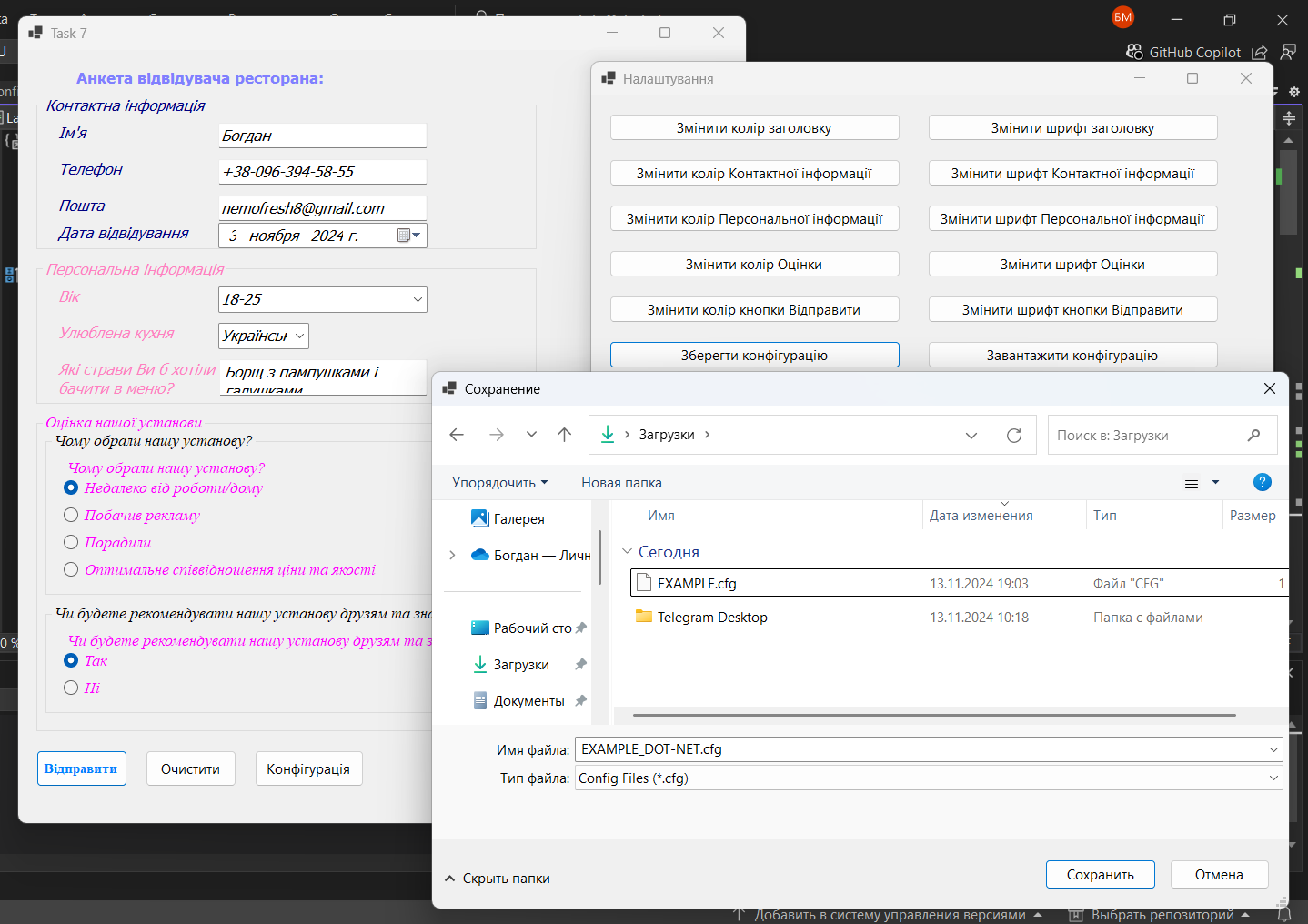
public FontStyle ButtonSubmitFontStyle { get; set; }

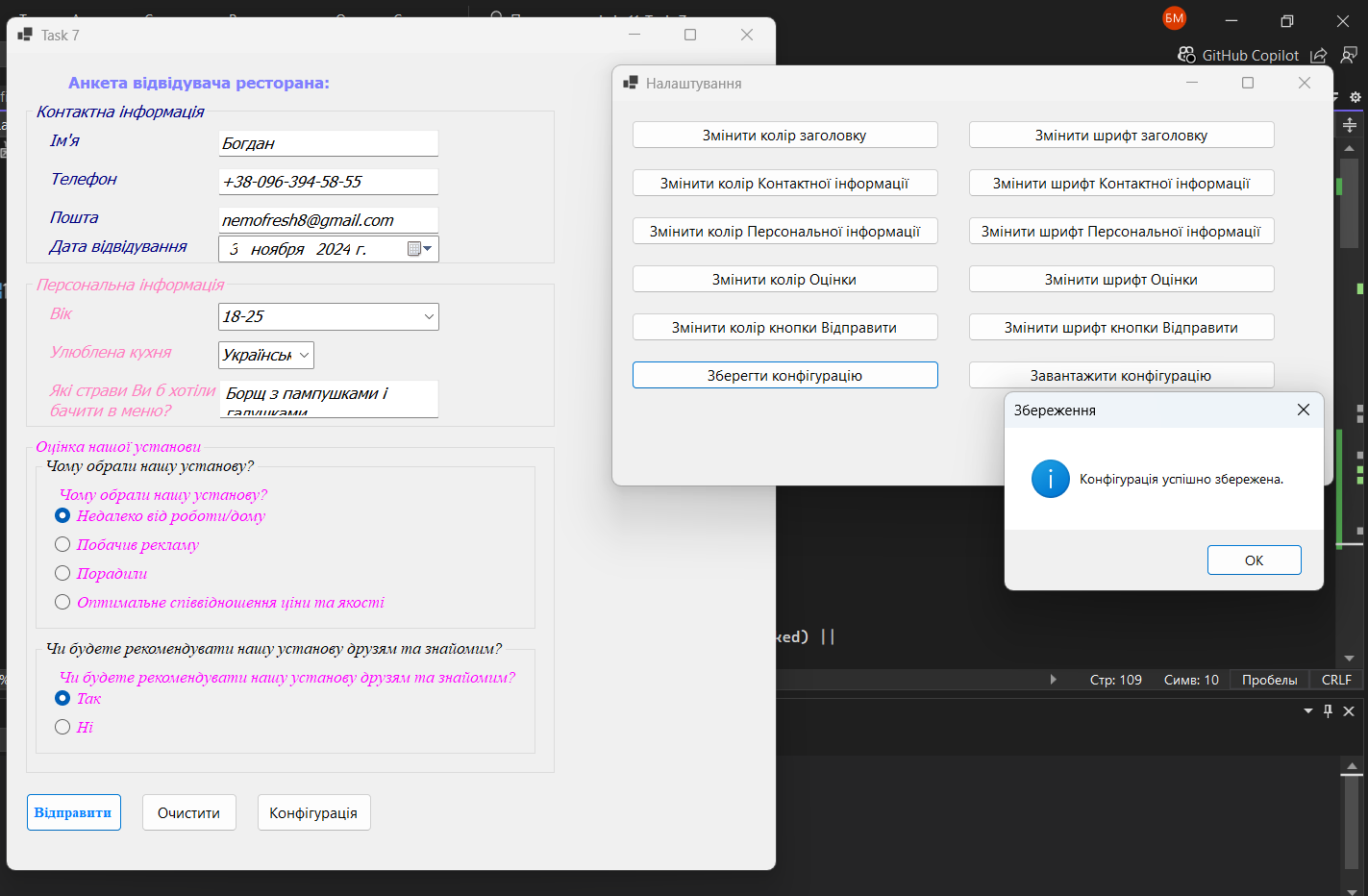
}

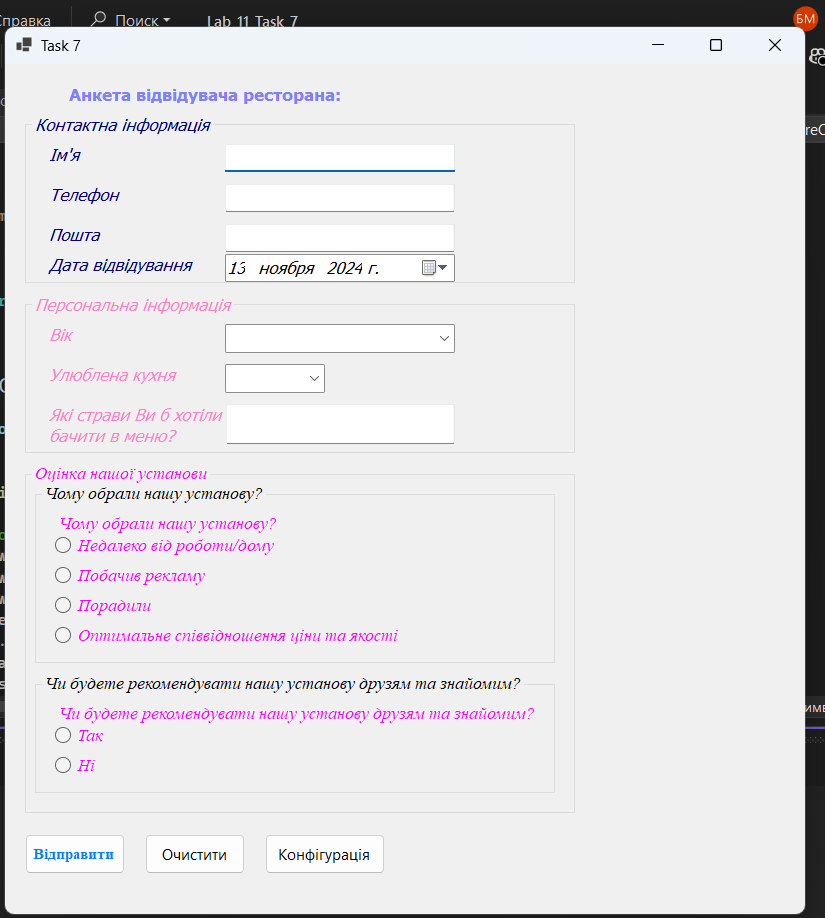
}

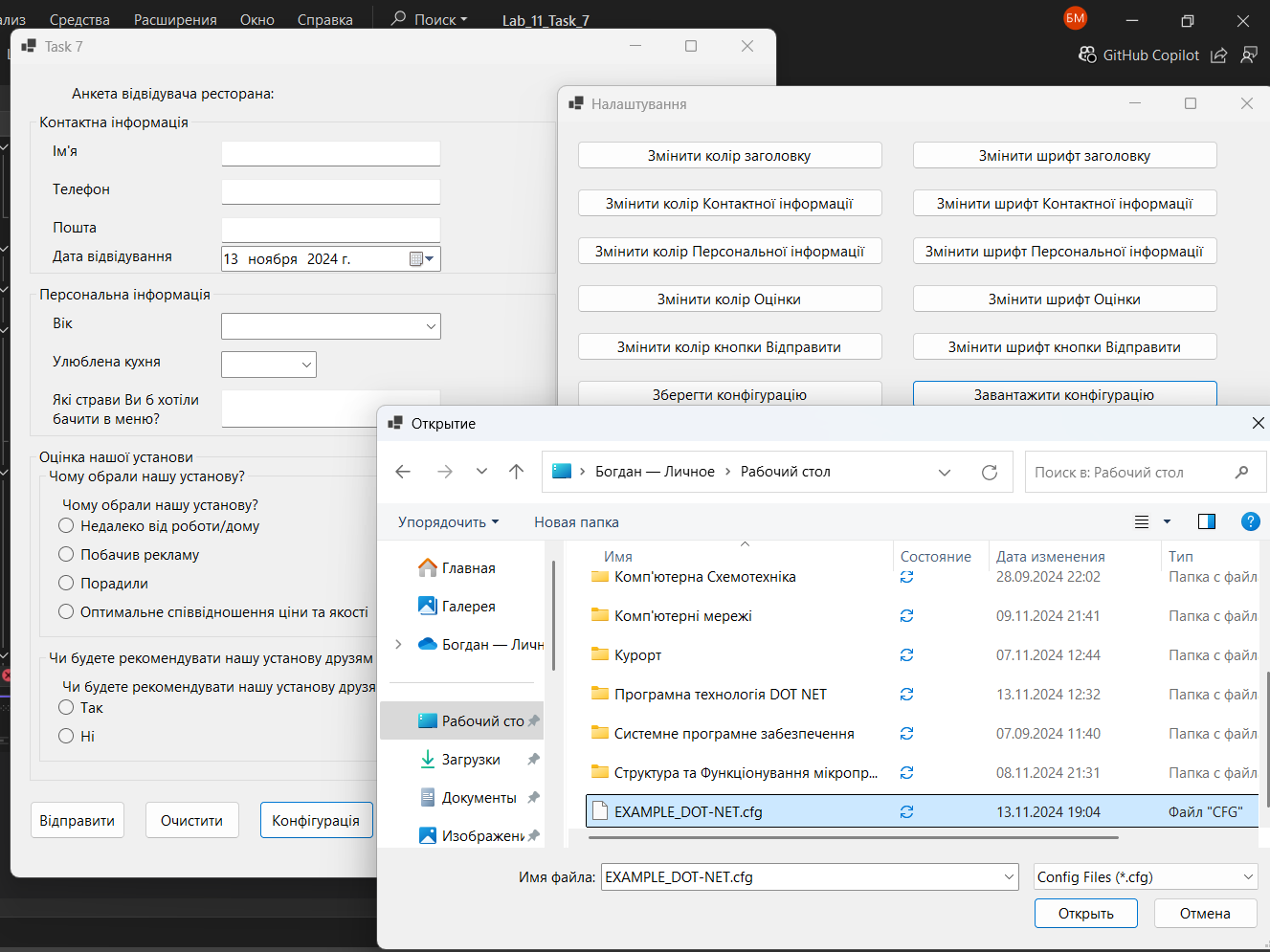
**Результат роботи програми:**

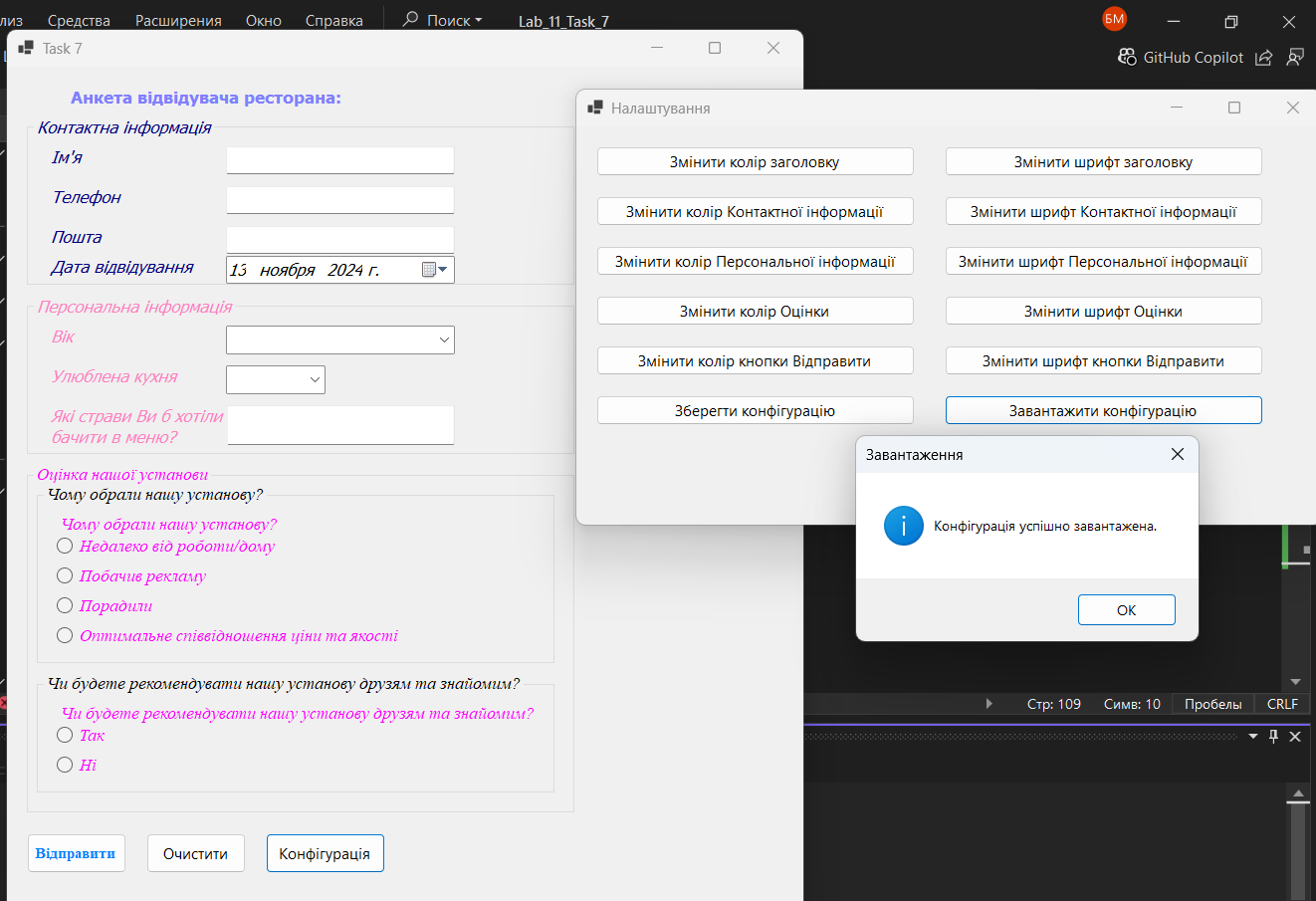












**Алгоритм побудови коду**

1. **Автоматичне завантаження конфігурації при запуску**:
   * У конструкторі MainForm додати завантаження налаштувань із файлу default.cfg, якщо він існує.
   * Створити метод LoadConfiguration, який зчитує налаштування з JSON файлу та застосовує їх до відповідних компонентів головної форми.
2. **Форма конфігурації (ConfigForm)**:
   * Додати кнопки для збереження та завантаження конфігурацій, що використовують SaveFileDialog і OpenFileDialog.
   * Метод buttonSaveConfig\_Click зберігає поточні параметри компонентів до вибраного файлу та default.cfg.
   * Метод buttonLoadConfig\_Click завантажує конфігурацію з вибраного файлу, використовуючи OpenFileDialog.
3. **Клас конфігурації**:
   * Створити клас Configuration, що містить властивості для збереження кольору, розміру шрифту, стилю шрифту для кожного з налаштованих компонентів.
   * Використовувати серіалізацію JSON для збереження та завантаження об’єкта Configuration у зовнішній файл.
4. **Застосування конфігурацій до компонентів**:
   * Методи ApplyConfiguration та SetFontAndColorConfig у класі MainForm забезпечують зчитування та застосування збережених параметрів до компонентів інтерфейсу.

**Функціональність**

* **Налаштування стилю компонентів**: Користувач може налаштовувати кольори та шрифти основних компонентів головної форми через конфігураційну форму.
* **Збереження та завантаження налаштувань**: Конфігурація може зберігатися у файл та завантажуватися з нього, що дозволяє легко відновити попередні налаштування.
* **Автоматичне завантаження налаштувань**: При запуску застосунку параметри автоматично завантажуються з файлу default.cfg, якщо він існує.

**Висновок**

Завдання виконано успішно, додано можливість зберігати та завантажувати налаштування інтерфейсу через конфігураційну форму. Це робить застосунок більш гнучким, дозволяючи користувачам легко зберігати та відновлювати бажані параметри інтерфейсу. Архітектура застосунку добре масштабована та дозволяє додавати нові налаштовувані параметри.

**Завдання 8**

Розробити застосунок текстовий редактор та забезпечити йому наступні функцїі:

* створення нового файлу, збереженні та завантаження файлу;
* виведення поточної позиції курсора у документі;
* вікно «Про програму» з ім'ям розробника програми.

Додатково доповнити функціями відповідно до варіанта

|  |  |
| --- | --- |
| № варіанту | Додаткова функція |
| 1 | 2 |
| 12. | Додати пункт меню, який дозволяе замінити всі числа в тексті на прописні аналоги |

**Текст програми:**

***MainForm.cs:***

using System;

using System.IO;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace Lab\_11\_Task\_8

{

public partial class MainForm : Form

{

private string currentFilePath = string.Empty;

public MainForm()

{

InitializeComponent();

richTextBox1.SelectionChanged += richTextBox1\_SelectionChanged;

}

// Створити новий файл

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

{

richTextBox1.Clear();

currentFilePath = string.Empty;

}

// Відкрити файл

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

{

using (OpenFileDialog ofd = new OpenFileDialog())

{

ofd.Filter = "Текстові файли (\*.txt)|\*.txt|Rich Text Files (\*.rtf)|\*.rtf|Усі файли (\*.\*)|\*.\*";

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

{

currentFilePath = ofd.FileName;

if (Path.GetExtension(currentFilePath).ToLower() == ".rtf")

{

richTextBox1.LoadFile(currentFilePath, RichTextBoxStreamType.RichText);

}

else

{

richTextBox1.LoadFile(currentFilePath, RichTextBoxStreamType.PlainText);

}

}

}

}

// Зберегти файл

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

{

if (!string.IsNullOrEmpty(currentFilePath))

{

SaveToFile(currentFilePath);

}

else

{

SaveAs();

}

}

// Зберегти як

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

{

SaveAs();

}

private void SaveAs()

{

using (SaveFileDialog sfd = new SaveFileDialog())

{

sfd.Filter = "Текстові файли (\*.txt)|\*.txt|Rich Text Files (\*.rtf)|\*.rtf|Усі файли (\*.\*)|\*.\*";

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

{

currentFilePath = sfd.FileName;

SaveToFile(currentFilePath);

}

}

}

private void SaveToFile(string filePath)

{

if (Path.GetExtension(filePath).ToLower() == ".rtf")

{

richTextBox1.SaveFile(filePath, RichTextBoxStreamType.RichText);

}

else

{

richTextBox1.SaveFile(filePath, RichTextBoxStreamType.PlainText);

}

}

// Відображення позиції курсора

private void richTextBox1\_SelectionChanged(object sender, EventArgs e)

{

int index = richTextBox1.SelectionStart;

int line = richTextBox1.GetLineFromCharIndex(index);

int column = index - richTextBox1.GetFirstCharIndexOfCurrentLine();

toolStripStatusLabel1.Text = $"Рядок: {line + 1}, Стовпець: {column + 1}";

}

// Показати "Про програму"

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

{

AboutForm aboutForm = new AboutForm();

aboutForm.ShowDialog();

}

// Замінити числа на слова

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

{

string originalText = richTextBox1.Text;

string newText = ReplaceNumbersWithWords(originalText);

richTextBox1.Text = newText;

}

private string ReplaceNumbersWithWords(string input)

{

return Regex.Replace(input, @"\d+", new MatchEvaluator(NumberToWordsEvaluator));

}

private string NumberToWordsEvaluator(Match match)

{

if (int.TryParse(match.Value, out int number))

{

return NumberToWords(number);

}

return match.Value;

}

private string NumberToWords(int number)

{

if (number == 0)

return "нуль";

if (number < 0)

return "мінус " + NumberToWords(Math.Abs(number));

string words = "";

if ((number / 1000000) > 0)

{

words += NumberToWords(number / 1000000) + " мільйон ";

number %= 1000000;

}

if ((number / 1000) > 0)

{

words += NumberToWords(number / 1000) + " тисяч ";

number %= 1000;

}

if ((number / 100) > 0)

{

words += NumberToWords(number / 100) + " сто ";

number %= 100;

}

if (number > 0)

{

if (words != "")

words += "";

var unitsMap = new[]

{

"", "один", "два", "три", "чотири", "п'ять",

"шість", "сім", "вісім", "дев'ять", "десять", "одинадцять",

"дванадцять", "тринадцять", "чотирнадцять", "п'ятнадцять",

"шістнадцять", "сімнадцять", "вісімнадцять", "дев'ятнадцять"

};

var tensMap = new[]

{

"", "десять", "двадцять", "тридцять", "сорок", "п'ятдесят",

"шістдесят", "сімдесят", "вісімдесят", "дев'яносто"

};

if (number < 20)

{

words += unitsMap[number];

}

else

{

words += tensMap[number / 10];

if ((number % 10) > 0)

words += " " + unitsMap[number % 10];

}

}

return words.Trim();

}

}

}

***AboutForm.cs:***

using System;

using System.Windows.Forms;

namespace Lab\_11\_Task\_8

{

public partial class AboutForm : Form

{

public AboutForm()

{

InitializeComponent();

}

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

{

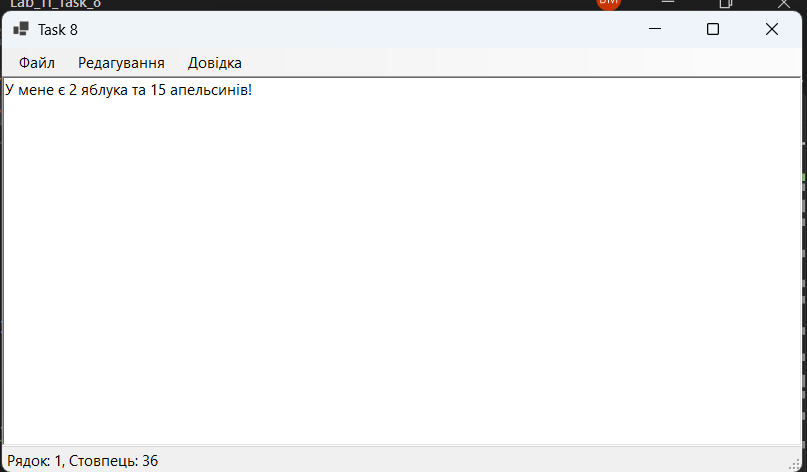
this.Close();

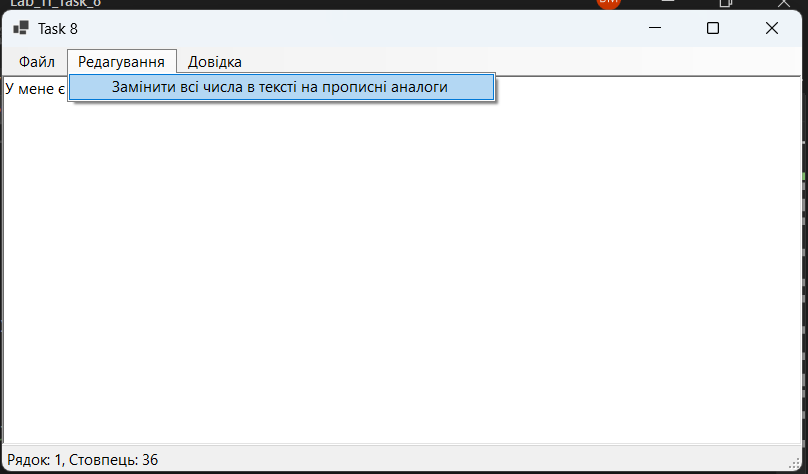
}

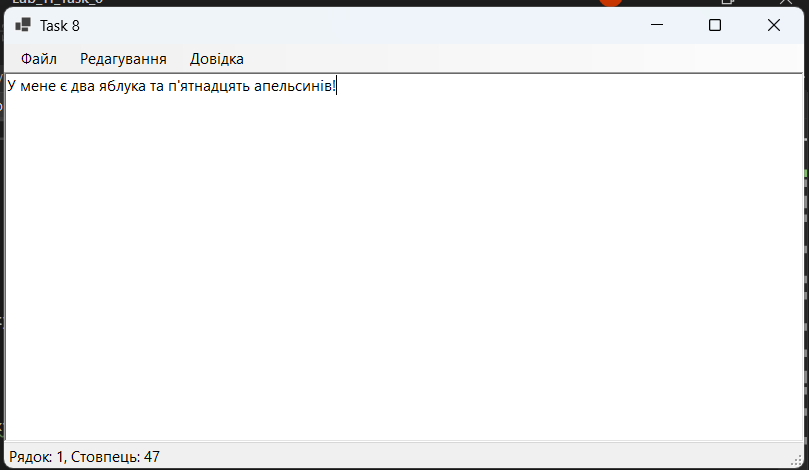
}

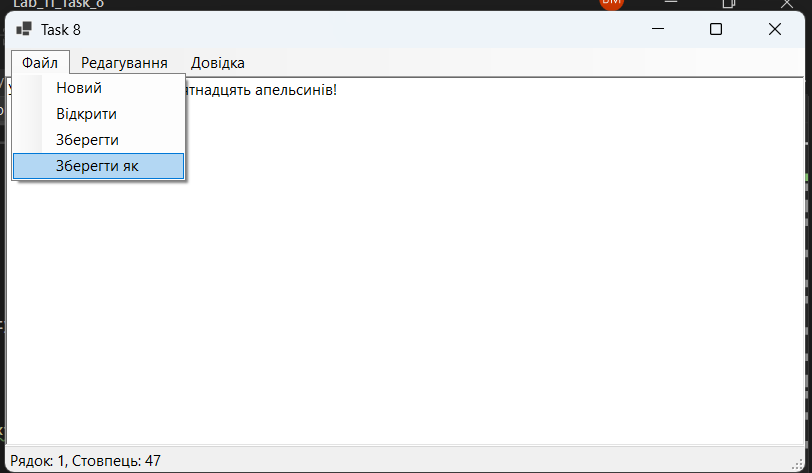
}

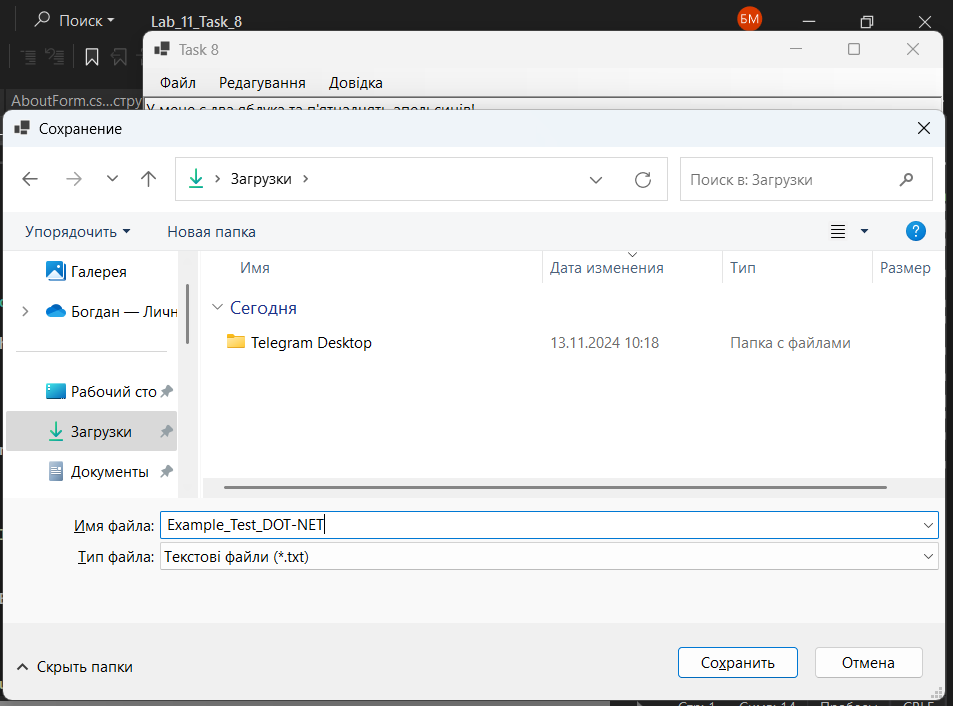
**Результат роботи програми:**

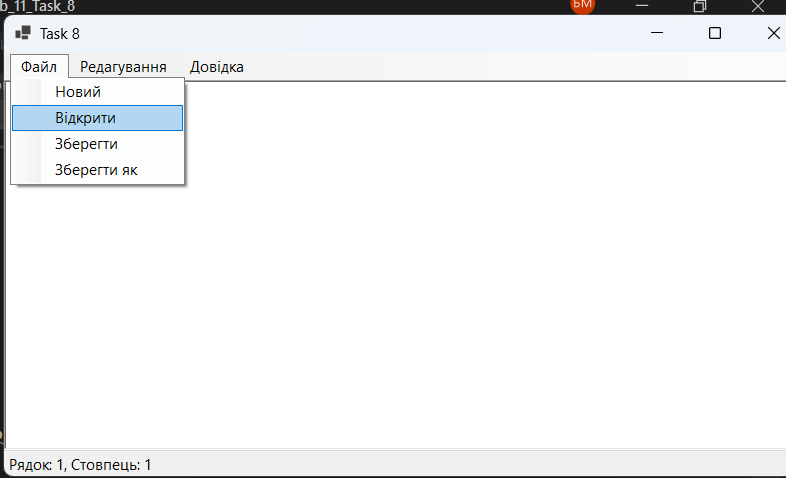


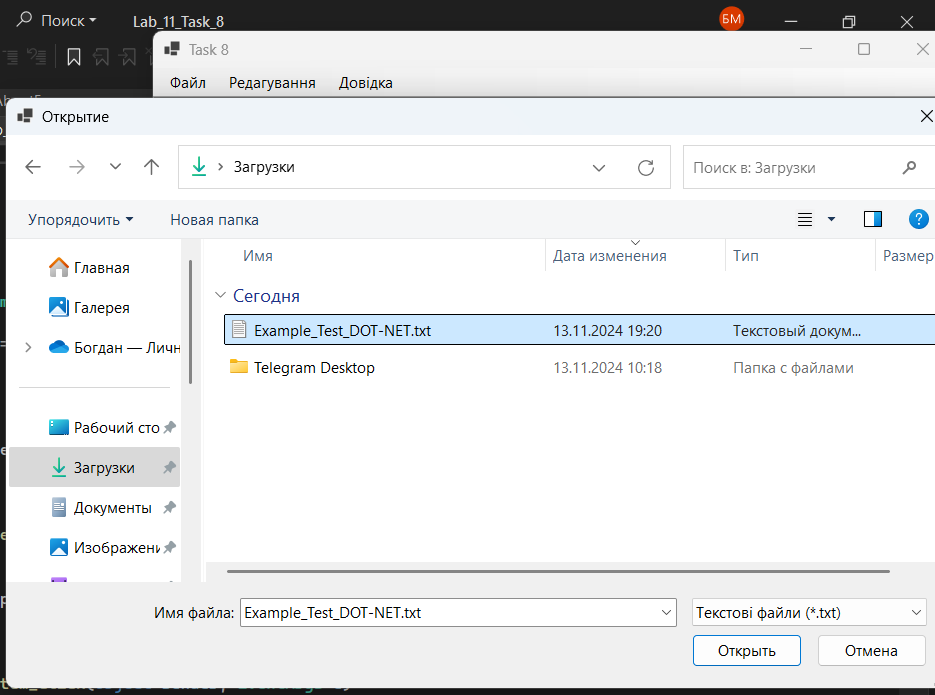


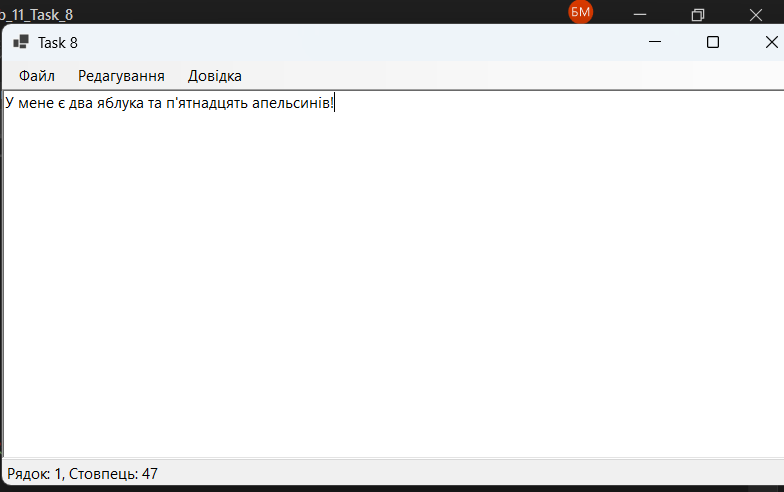


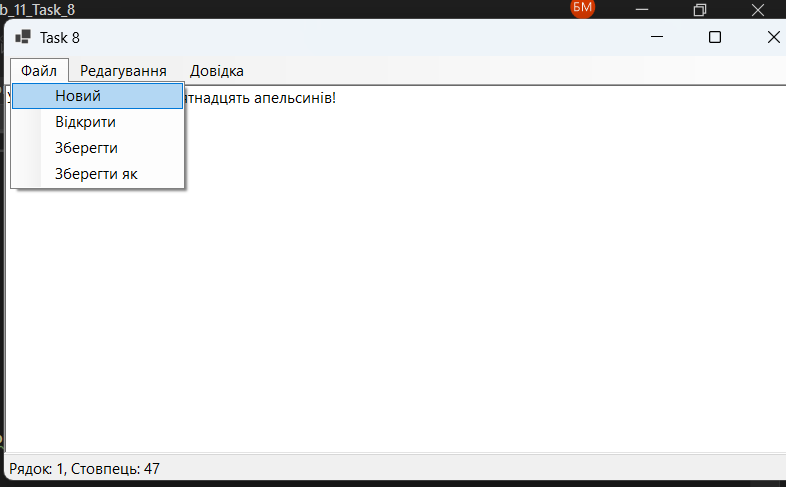


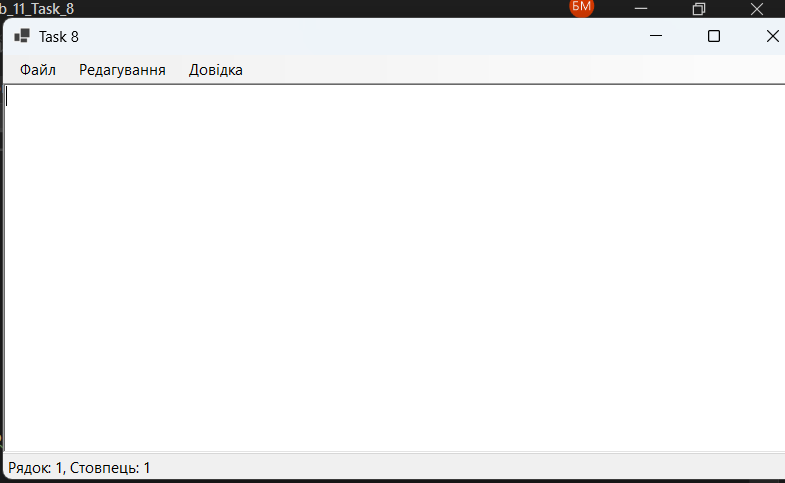


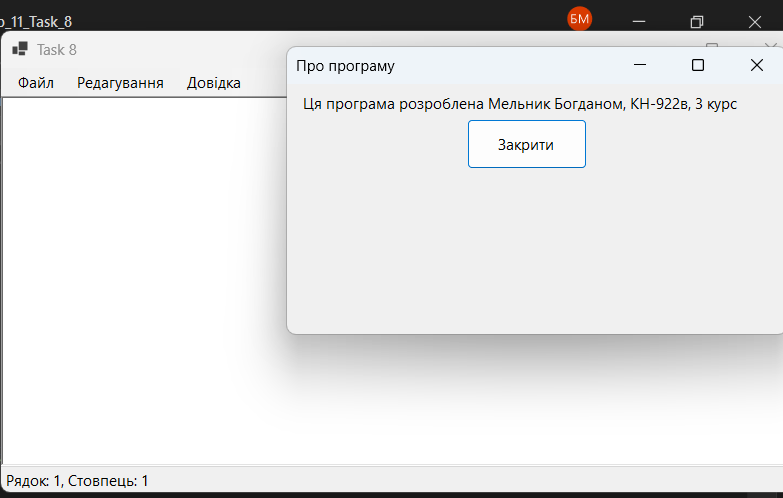












**Алгоритм побудови коду**

1. **Ініціалізація та підключення подій**:
   * У класі MainForm було створено обробник для зміни позиції курсора richTextBox1\_SelectionChanged, який забезпечує оновлення статусу з поточною позицією курсора.
2. **Основні функції текстового редактора**:
   * Створено функції для **створення нового файлу**, **збереження** та **відкриття файлу** з використанням відповідних діалогових вікон (SaveFileDialog та OpenFileDialog). Файли можуть зберігатись як у форматі txt, так і rtf.
   * При відкритті або збереженні файлу визначається тип файлу, щоб використовувати правильний формат (PlainText або RichText).
3. **Відображення позиції курсора**:
   * Метод richTextBox1\_SelectionChanged обчислює поточний рядок і стовпець курсора в richTextBox1 і оновлює toolStripStatusLabel1.
4. **Діалог «Про програму»**:
   * Створено форму AboutForm, яка відкривається з пункту меню "Про програму". У ній відображається інформація про розробника.
5. **Замінник чисел на текстові аналоги**:
   * Метод replaceNumbersWithWordsToolStripMenuItem\_Click обробляє заміну чисел у тексті на прописні аналоги, використовуючи регулярні вирази (Regex.Replace) та функцію NumberToWords, яка конвертує числа в текстовий формат.

**Функціональність**

* **Створення нового файлу**: очищує richTextBox1 та скидає шлях до файлу.
* **Збереження/Збереження як**: зберігає текст у файл з вибором формату.
* **Завантаження файлу**: відкриває текстовий файл у редакторі з підтримкою txt і rtf форматів.
* **Відображення позиції курсора**: дозволяє користувачу бачити поточне розташування курсора.
* **Про програму**: відображає вікно з інформацією про розробника.
* **Заміна чисел**: пункт меню дозволяє автоматично змінити числа на текстові аналоги у всьому документі.

**Висновок**

Застосунок реалізує базовий функціонал текстового редактора з можливістю створення, збереження та завантаження файлів. Окрім того, додано додаткову функцію заміни чисел на текстові аналоги, що відповідає варіанту №12. Це забезпечує користувачам зручність у редагуванні та форматуванні тексту, а також відображає поточну позицію курсора, що є корисним для більш точного редагування документів.

**Загальний висновок**

Під час виконання лабораторної роботи №11 було розроблено низку застосунків з багато віконним інтерфейсом, які демонструють різні способи організації взаємодії між формами, включаючи передачу даних, налаштування стилів, обчислення та обробку введених даних.

**Основні результати:**

* Створено застосунок з переходами між формами згідно заданої граф-схеми, де кожна форма має унікальні кольорові та шрифтові характеристики.
* Реалізовано обмін даними між формами, дозволяючи передавати інформацію в обидва напрямки, що покращує контекстну обізнаність користувача.
* Розроблено чотири варіанти застосунку для обчислення геометричних параметрів трикутника, що демонструють різні способи введення та відображення даних.
* Створено чотири варіанти застосунку для розрахунку математичного виразу, з різними конфігураціями форм для введення та виведення результатів.
* Розроблено застосунок для вибору міста з випадаючого списку, з відображенням фото та опису міста на додатковій формі.
* Додано форму конфігурації для налаштування кольору та шрифту компонентів головної форми, з можливістю збереження та завантаження конфігурацій з файлу.
* Створено застосунок текстового редактора з базовим функціоналом: створення, збереження, відкриття файлів, відображення позиції курсора, вікно "Про програму" та додаткову функцію заміни чисел на прописні аналоги.

**Висновки:**

* Засвоєно методи створення та керування багато віконними застосунками в C#.
* Отримано практичний досвід у передачі даних між формами, обробці введених даних та налаштуванні стилів компонентів.
* Реалізовано різні варіанти організації інтерфейсу для вирішення однієї задачі, що демонструє гнучкість у розробці застосунків.
* Застосовано компоненти OpenFileDialog та SaveFileDialog для роботи з файлами, що розширює можливості застосунків.
* Зроблено висновок про важливість багато віконних застосунків для створення зручних та функціональних програм.