Міністерство освіти і науки України

Хмельницький політехнічний фаховий коледж

Національного університету “Львівська політехніка”

**Відділення програмної інженерії**

Циклова комісія

Програмного забезпечення

**Звіт**

з навчальної практики з програмування

Виконав студент ***II*** курсу, групи ***ПІ-233***

Спеціальність ***121 “Інженерія програмного забезпечення”***

***Дар’єв Даниїл Олександрович***

Керівник **Валентина ГУМЕННА**

Члени комісії

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Підпис) (Прізвище та ініціали)

Денис Жереб

(Підпис) (Прізвище та ініціали)

Валентина Гуменна

(Підпис) (Прізвище та ініціали)

Оцінка

дата захисту

Хмельницький, 2025

**Зміст**

[Форма з логіном , паролем та реєстрацією 4](#_Toc201580365)

[Головна (навігаційна) форма 12](#_Toc201580366)

[Завдання 1 Обчислити площу поверхні та об’єм зрізаного конуса 16](#_Toc201580367)

[Завдання 2 Табуляція функції 20](#_Toc201580368)

[Завдання 3 Виконання дій над масивом 32](#_Toc201580369)

[Завдання 4 Побудова графічних зображень на основі математичних функцій. 41](#_Toc201580370)

[Завдання 5 Робота зі структурою 48](#_Toc201580371)

[Висновок 60](#_Toc201580372)

Створення програм із графічним інтерфейсом в Visual Studio C# Windows Form

***Мета*:** Навчитись працювати із графічним інтерфейсом Microsoft Windows Form та створювати в ньому програми.

***Умова завдання:***

Програма має містити форми для п’ятьох завдань, кожне із завдань має власну умову та взаємодію між собою через головну форму, для взаємодії із головною формою необхідно ввести логін та пароль

***Теоретичні відомості:***

1. Form – фоновий компонент проекту, який при запуску створює нове вікно, де розташовуються інші об’єкти для подальшого використання.
2. Label – компонет, що слугує для розміщення тексту на екрані без можливості його змінити користувачу.
3. Button – компонент, що використовується зазвичай для запуску певних подій при події Click.
4. CheckBox – компонент, що може відображати два значення: правда (true) або брехня (false).
5. DataGridVeiw – таблиця, з якою може працювати користувач, в програмі, опрацьовуються рядки і стовпці. Доцільно використовувати в базах даних.
6. TextBox – поле для введення даних користувачем. За допомогою події KeyPress можемо перевіряти коректність введених даних.
7. ContextMenuStrip – це спливаюче меню, яке зазвичай з'являється при натисканні ПКМ на елементі керування або формі.
8. MenuStrip – елемент керування закріплений зверху форми, з яким можна взаємодіяти.
9. ToolStrip – елемент керування, для розміщення інших елементів, подібний до MenuStrip, тільки із більшим функціоналом.
10. OpenFileDialog - відкриває стандартне діалогове вікно, за допомогою якого користувач може відкрити файл.
11. SaveFileDialog - пропонує користувачеві вибрати місце для збереження файлу.
12. TabControl – компонент, що містить вкладки, між якими можна переміщуватися і які місять інші компоненти.
13. Panel – контейнер, в якому можна розміщувати елементи керування.
14. PictureBox – елемент керування, для розміщення зображень.
15. ComboBox – елемент керування, поданий у вигляді списку.
16. Timer – елемент керування, який має подію, що оновлюється через вказаний інтервал.
17. StatusStrip – елемент керування, розміщений знизу форми, для відображення певної інформації.
18. Chart – елемент керування, для відображення інформації у вигляді діаграми/графіка.
19. GroupBox – контейнер, для розміщення та групування елементів.
20. RadioButton – елемент керування, у вигляді кнопки перемикання.

***Хід роботи:***

Завантажено Microsoft Visual Studio. Створено проєкт .Windows Form (.Net 9.0)ІПЗ\_2025\_Дар’єв

Форма з логіном , паролем та реєстрацією

1. За допомогою Панелі інструментів було створено такі об’єкти як Button, Label, TextBox та розміщено для зручної взаємодії(див. Рисунок 1).

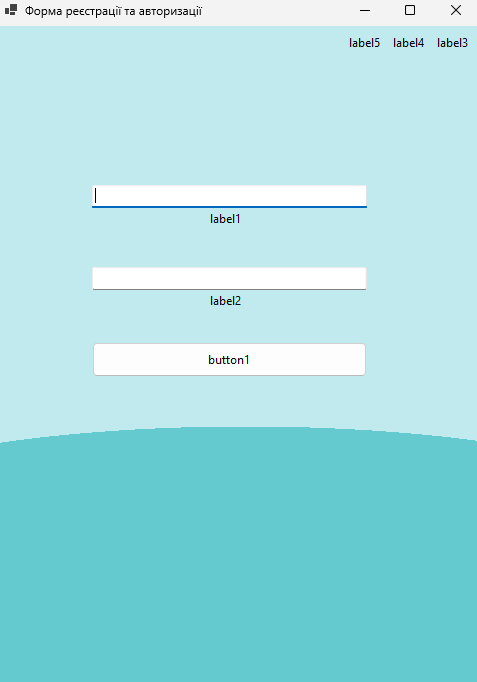


Рисунок 1 - Пуста форма входу

1. Додано надписи.Змінено колір,шрифт та розмір за допомогою інспектора об’єктів та властивостей. Сформовано кінцевий вигляд вікна (див. Рисунок 2).



Рисунок 2 - Кінцевий варіант форми входу та реєстрації

1. Написано програмний код для функціонування форми в класі RegistrationForm в просторі імен ІндивідуальнеЗавданняCSharp

#region Змінні для дизайна форми

private bool wasHiddenManually = false;

private bool dragging = false;

private Point dragCursorPoint;

private Point dragFormPoint;

#endregion

#region Змінні для шифрування

public struct UserEntry

{

public string userLogin;

public string userEncryptedPassword;

}

private readonly string \_filePath = "userEntryDataBase.data";

private List<UserEntry> \_users = new List<UserEntry>();

#endregion

#region Події в формі реєстрації

public RegistrationForm()

{

InitializeComponent();

userLoad();

string relativePath = Path.Combine("Icons", "eye\_on.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

passwordPictureBox.Image = svgDoc.Draw();

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_RESTORE = 0xF120;

if (m.Msg == WM\_SYSCOMMAND && m.WParam.ToInt32() == SC\_RESTORE)

{

if (wasHiddenManually)

{

wasHiddenManually = false;

this.WindowState = FormWindowState.Normal;

Rectangle screen = Screen.FromControl(this).WorkingArea;

int x = (screen.Width - this.Width) / 2 + screen.Left;

int y = (screen.Height - this.Height) / 2 + screen.Top;

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

this.Opacity = 1;

this.Enabled = true;

}

}

base.WndProc(ref m);

}

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

{

using (Brush paintBrush = new SolidBrush(Color.FromArgb(101, 201, 208)))

{

e.Graphics.FillRectangle(paintBrush, new RectangleF(0F, 500F, 500F, 200F));

e.Graphics.FillEllipse(paintBrush, new RectangleF(-200F, 400F, 900F, 200F));

}

}

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

{

Application.Exit();

}

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

{

wasHiddenManually = true;

this.WindowState = FormWindowState.Minimized;

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = true;

dragCursorPoint = Cursor.Position;

dragFormPoint = this.Location;

}

}

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

{

if (dragging)

{

Point diff = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));

this.Location = Point.Add(dragFormPoint, new Size(diff));

}

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = false;

}

}

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

{

if (passwordTextBox.UseSystemPasswordChar == true)

{

string relativePath = Path.Combine("Icons", "eye\_off.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

passwordPictureBox.Image = svgDoc.Draw();

passwordTextBox.UseSystemPasswordChar = false;

}

else

{

string relativePath = Path.Combine("Icons", "eye\_on.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

passwordPictureBox.Image = svgDoc.Draw();

passwordTextBox.UseSystemPasswordChar = true;

}

}

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

{

if (!char.IsDigit(e.KeyChar) && !char.IsLetter(e.KeyChar) && e.KeyChar != '\b')

{

e.Handled = true;

}

}

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

{

if (!char.IsDigit(e.KeyChar) && !char.IsLetter(e.KeyChar) && e.KeyChar != '\b')

{

e.Handled = true;

}

}

private void label3\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(loginTextBox.Text) || string.IsNullOrEmpty(passwordTextBox.Text))

{

errorInputLabel.Visible = true;

return;

}

try

{

userRegister(loginTextBox.Text, passwordTextBox.Text);

}

catch (Exception ex)

{

errorInputLabel.Location = new Point(171, 355);

errorInputLabel.Visible = true;

errorInputLabel.Text = ex.Message;

}

}

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

{

bool success = userAuthenticate(loginTextBox.Text, passwordTextBox.Text);

if (!success)

{

errorInputLabel.Location = new Point(155, 355);

errorInputLabel.Text = "Невірний логін або пароль.";

errorInputLabel.Visible = true;

return;

}

this.Hide();

MainMenuWindow mainMenu = new MainMenuWindow();

mainMenu.Show();

mainMenu.Activate();

}

#endregion

#region Шифрування да дешифрування пароля

public static string Encrypt(string input, int shift)

{

StringBuilder result = new StringBuilder();

foreach (char c in input)

{

if (char.IsLetter(c))

{

char baseChar = char.IsUpper(c) ? 'A' : 'a';

char shifted = (char)(((c - baseChar + shift) % 26) + baseChar);

result.Append(shifted);

}

else

{

result.Append(c);

}

}

return result.ToString();

}

public static string Decrypt(string input, int shift)

{

return Encrypt(input, 26 - (shift % 26));

}

private int userGetShift(string login) { return login.Length % 26; }

#endregion

#region Операції з користувачем

private void userLoad()

{

\_users = new List<UserEntry>();

if (!File.Exists(\_filePath))

return;

using (var fs = new FileStream(\_filePath, FileMode.Open, FileAccess.Read))

using (var reader = new BinaryReader(fs, Encoding.UTF8))

{

int count = reader.ReadInt32();

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

{

string login = reader.ReadString();

string encryptedPassword = reader.ReadString();

\_users.Add(new UserEntry { userLogin = login, userEncryptedPassword = encryptedPassword });

}

}

}

private void userSave()

{

using (var fs = new FileStream(\_filePath, FileMode.Create, FileAccess.Write))

using (var writer = new BinaryWriter(fs, Encoding.UTF8))

{

writer.Write(\_users.Count);

foreach (var user in \_users)

{

writer.Write(user.userLogin);

writer.Write(user.userEncryptedPassword);

}

}

}

public void userRegister(string login, string password)

{

bool userExists = false;

foreach (UserEntry u in \_users)

{

if (u.userLogin == login)

{

userExists = true;

break;

}

}

if (userExists)

throw new Exception("Користувач вже існує.");

int shift = userGetShift(login);

string encryptedPassword = Encrypt(password, shift);

\_users.Add(new UserEntry { userLogin = login, userEncryptedPassword = encryptedPassword });

userSave();

}

public bool userAuthenticate(string login, string password)

{

int shift = userGetShift(login);

string encryptedInput = Encrypt(password, shift);

var user = \_users.Find(u => u.userLogin == login);

if (user.userLogin == null)

return false;

return user.userEncryptedPassword == encryptedInput;

}

#endregion

1. Перевірено функціонування програмного коду

Перевірено поведінку форми при введені логіну і паролю(див. Рисунок 3)

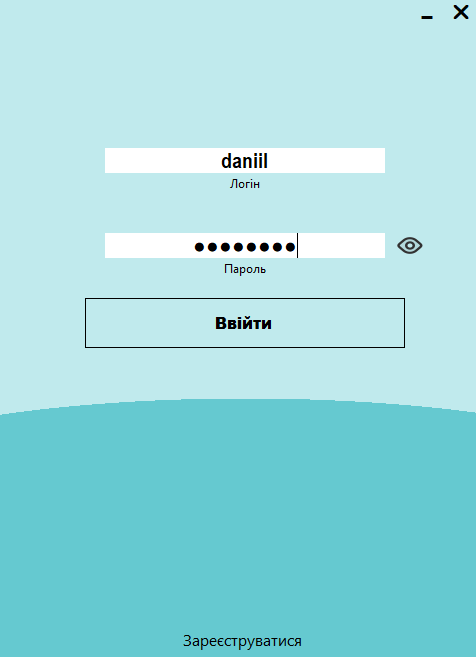


Рисунок 3 - Поведінка при введені данних

Перевірено поведінку форми при натисканні кнопки для відображення паролю(див. Рисунок 4)

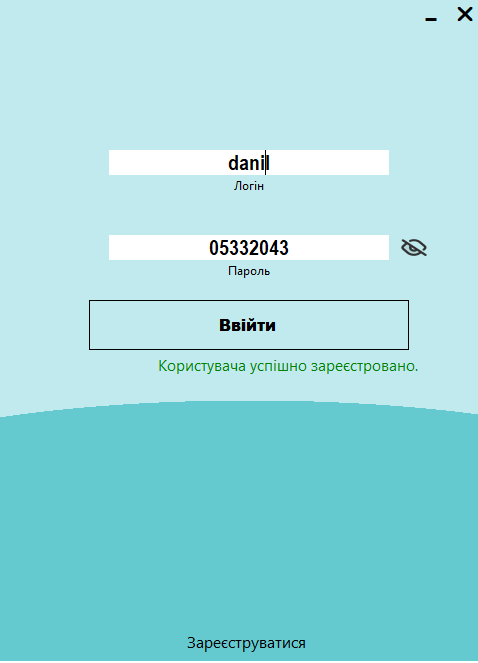


Рисунок 4 - Перевірка кнопки відображення паролю

Перевірка на невірно введений логін або пароль(див. Рисунок 5)

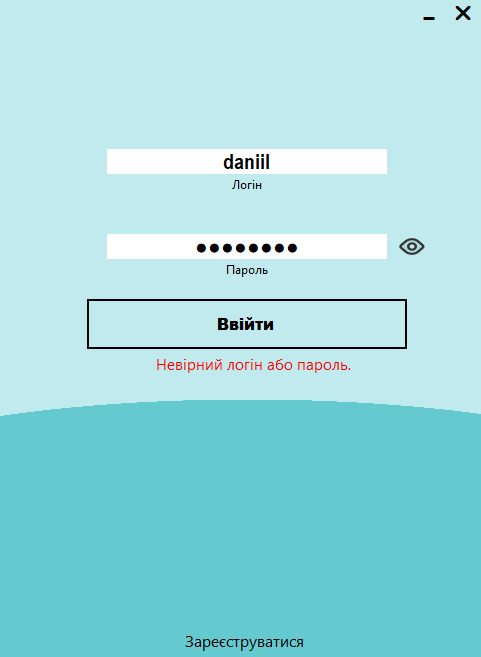


Рисунок 5 - Неправильний логін або пароль

Перевірка функціонування кнопки Зареєструватися(див. Рисунок 6)

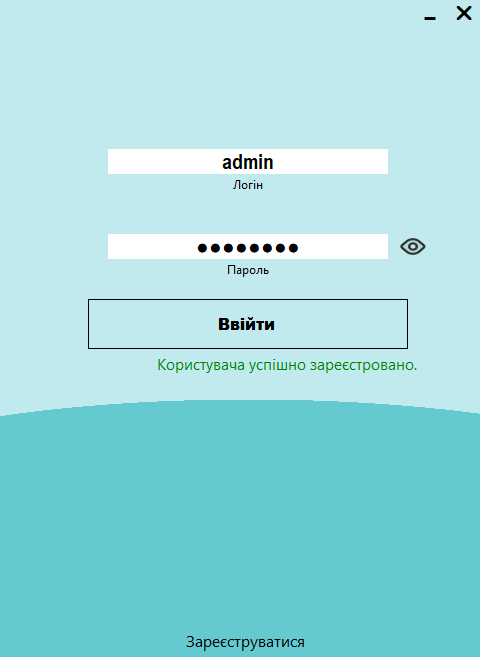


Рисунок 6 - Успішна реєстрація

Перевірка якщо зареєстрований користувач вже існує(див. Рисунок 7)

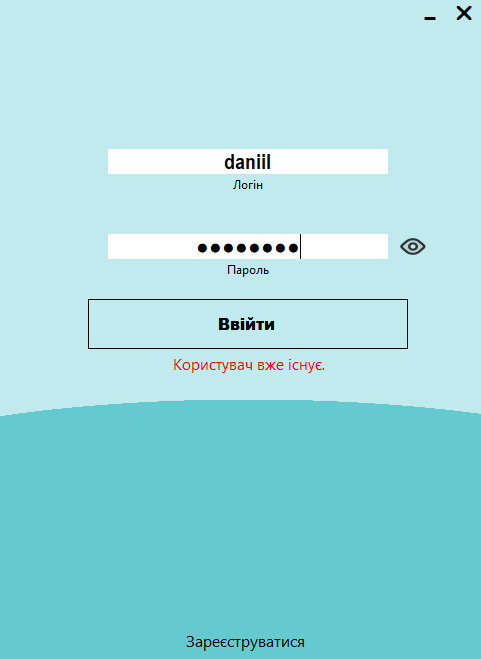


Рисунок 7 - Виведення помилки що зареєстрований користувач вже існує

Головна (навігаційна) форма

1. Створено нову форму та додано до неї такі елементи як: MenuStrip,GroupBox,Label,ToolStripMenuItem (див. Рисунок 8)

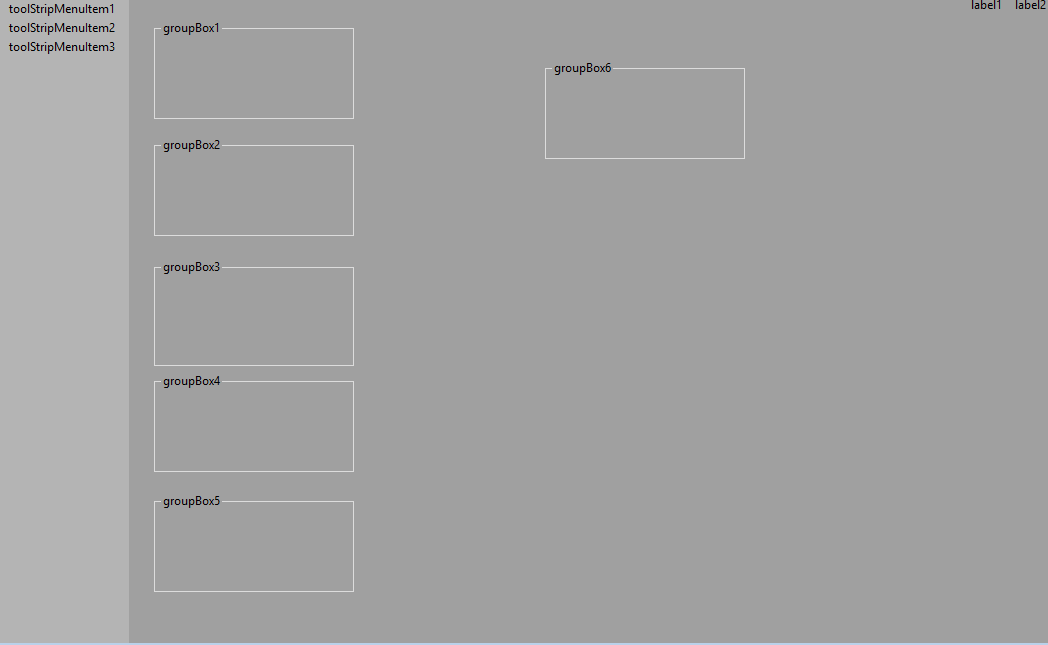


Рисунок 8 - Проектування дизайну головної форми

1. Налаштовано розмір/колір/розташування форми та її вмісту(див. Рисунок 9/10/11)

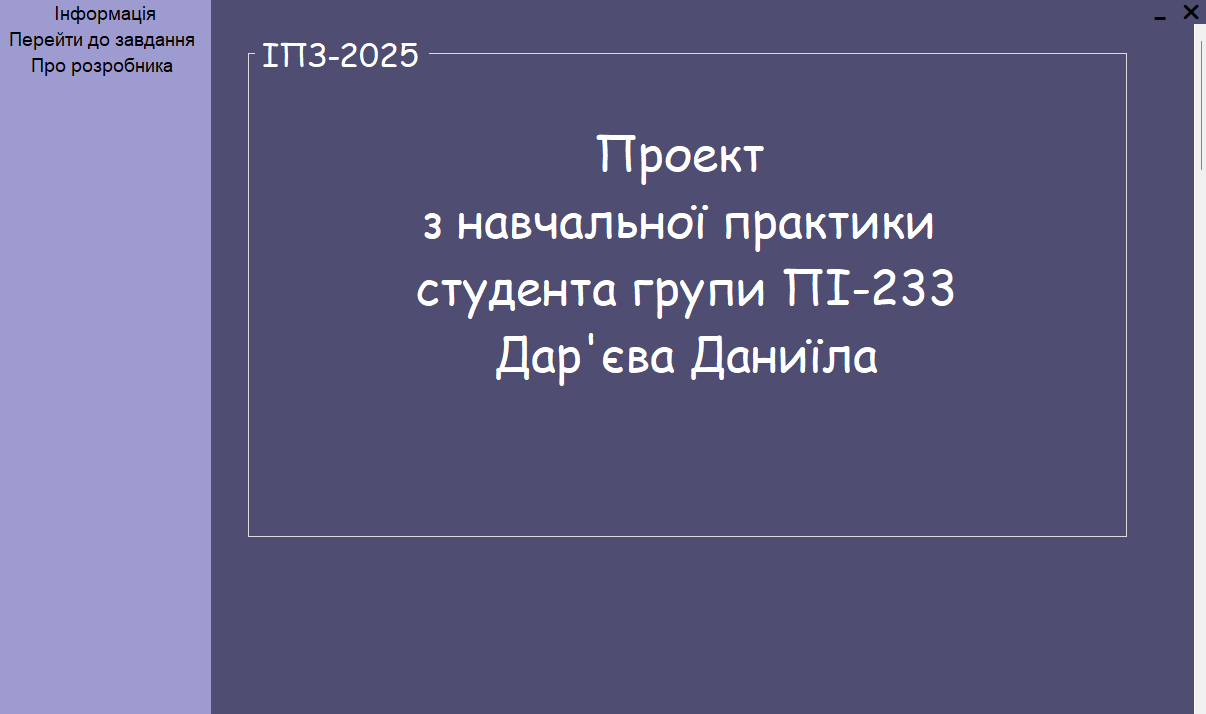


Рисунок 9 - Інформація про розробника на головній формі

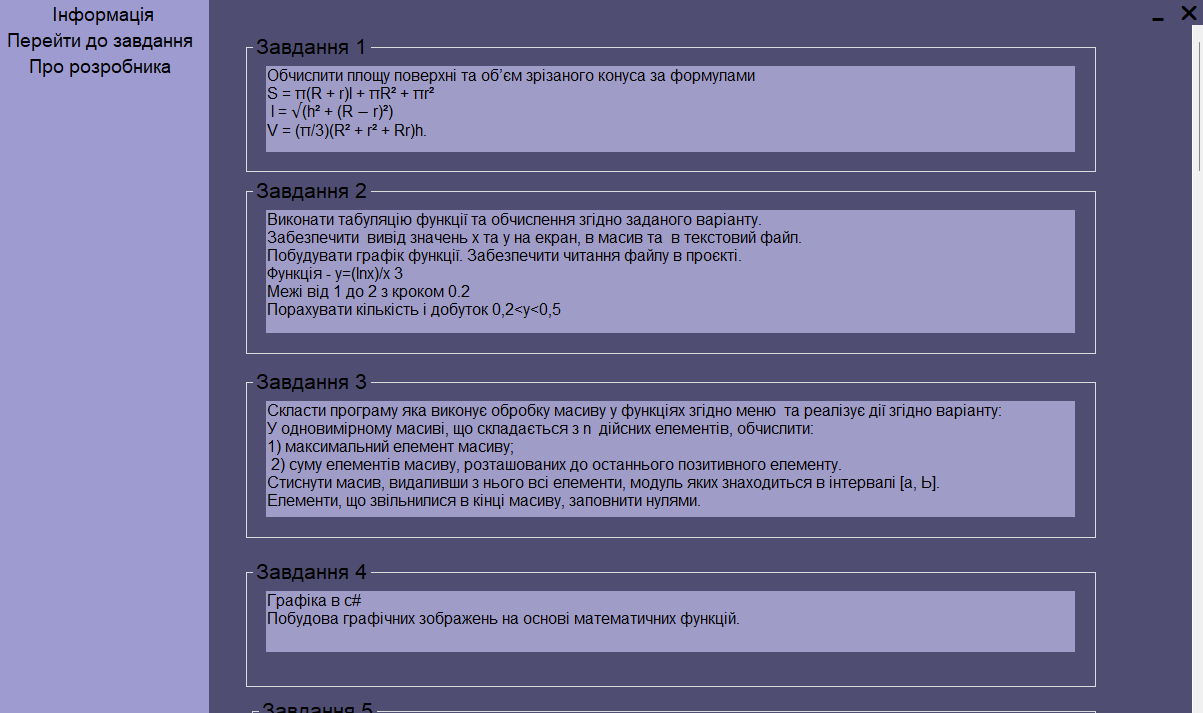


Рисунок 10 - Інформація про завдання

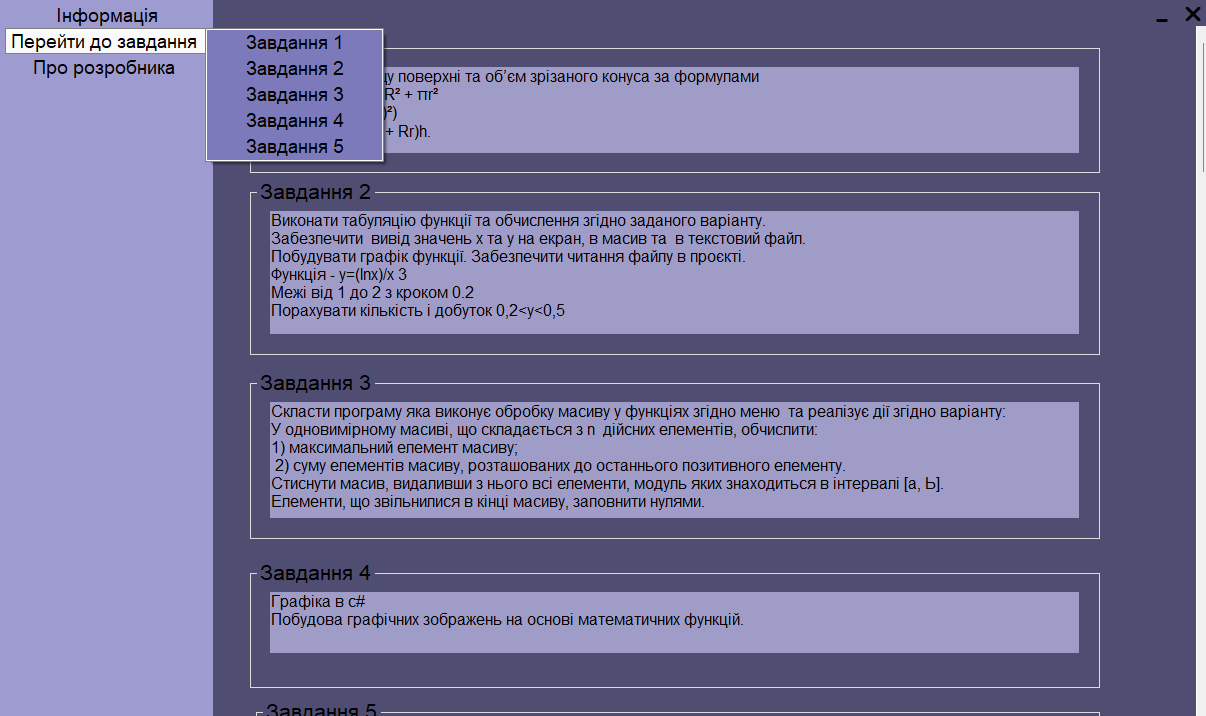


Рисунок 11 - Кнопки навігації по завданнях

1. Створено відповідний програмний код для цієї форми в просторі імен ІндивідуальнеЗавданняCSharp

private bool wasHiddenManually = false;

private bool dragging = false;

private Point dragCursorPoint;

private Point dragFormPoint;

public MainMenuWindow()

{

InitializeComponent();

this.Activate();

}

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

{

Application.Exit();

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_RESTORE = 0xF120;

if (m.Msg == WM\_SYSCOMMAND && m.WParam.ToInt32() == SC\_RESTORE)

{

if (wasHiddenManually)

{

wasHiddenManually = false;

this.WindowState = FormWindowState.Normal;

Rectangle screen = Screen.FromControl(this).WorkingArea;

int x = (screen.Width - this.Width) / 2 + screen.Left;

int y = (screen.Height - this.Height) / 2 + screen.Top;

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

this.Opacity = 1;

this.Enabled = true;

}

}

base.WndProc(ref m);

}

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

{

wasHiddenManually = true;

this.WindowState = FormWindowState.Minimized;

}

private void infoMenuScrollBar\_Scroll(object sender, ScrollEventArgs e)

{

int delta = 10;

if (e.OldValue >= e.NewValue)

{

if (taskOneGroupBox.Location.Y > 26)

{

return;

}

taskOneGroupBox.Location = new Point(taskOneGroupBox.Location.X, taskOneGroupBox.Location.Y + delta);

taskTwoGroupBox.Location = new Point(taskTwoGroupBox.Location.X, taskTwoGroupBox.Location.Y + delta);

taskThreeGroupBox.Location = new Point(taskThreeGroupBox.Location.X, taskThreeGroupBox.Location.Y + delta);

taskFourGroupBox.Location = new Point(taskFourGroupBox.Location.X, taskFourGroupBox.Location.Y + delta);

taskFiveGroupBox.Location = new Point(taskFiveGroupBox.Location.X, taskFiveGroupBox.Location.Y + delta);

}

else

{

taskOneGroupBox.Location = new Point(taskOneGroupBox.Location.X, taskOneGroupBox.Location.Y - delta);

taskTwoGroupBox.Location = new Point(taskTwoGroupBox.Location.X, taskTwoGroupBox.Location.Y - delta);

taskThreeGroupBox.Location = new Point(taskThreeGroupBox.Location.X, taskThreeGroupBox.Location.Y - delta);

taskFourGroupBox.Location = new Point(taskFourGroupBox.Location.X, taskFourGroupBox.Location.Y - delta);

taskFiveGroupBox.Location = new Point(taskFiveGroupBox.Location.X, taskFiveGroupBox.Location.Y - delta);

}

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = true;

dragCursorPoint = Cursor.Position;

dragFormPoint = this.Location;

}

}

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

{

if (dragging)

{

Point diff = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));

this.Location = Point.Add(dragFormPoint, new Size(diff));

}

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = false;

}

}

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

taskOneGroupBox.Visible = true;

taskTwoGroupBox.Visible = true;

taskThreeGroupBox.Visible = true;

taskFourGroupBox.Visible = true;

taskFiveGroupBox.Visible = true;

infoMenuScrollBar.Visible = true;

AboutTheDeveloperGroupBox.Visible = false;

}

private void toolStripMenuItem4\_Click(object sender, EventArgs e)

{

AboutTheDeveloperGroupBox.Visible = true;

taskOneGroupBox.Visible = false;

taskTwoGroupBox.Visible = false;

taskThreeGroupBox.Visible = false;

taskFourGroupBox.Visible = false;

taskFiveGroupBox.Visible = false;

infoMenuScrollBar.Visible = false;

}

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

{

this.Hide();

TaskOneWindow taskOne = new TaskOneWindow();

taskOne.Show();

taskOne.Activate();

}

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

{

this.Hide();

TaskTwoWindow taskTwo = new TaskTwoWindow();

taskTwo.Show();

taskTwo.Activate();

}

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

{

this.Hide();

TaskThreeWindow taskThree = new TaskThreeWindow();

taskThree.Show();

taskThree.Activate();

}

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

{

this.Hide();

TaskFiveWindow taskFive = new TaskFiveWindow();

taskFive.Show();

taskFive.Activate();

}

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

{

this.Hide();

TaskFourWindow taskFour = new TaskFourWindow();

taskFour.Show();

taskFour.Activate();

}

}

Завдання 1 Обчислити площу поверхні та об’єм зрізаного конуса

1. Створено нову форму з назвою TaskOneWindow та додано різні елементи керування (див. Рисунок 12) необхідні для виконання завдання:

**Обчислити площу поверхні та об’єм зрізаного конуса за формулами**

**l = √(h² + (R − r)²)**

**S = π(R + r)l + πR² + πr²**

**V = (π/3)(R² + r² + Rr)h**

**Значення радіусів основ зрізаного конуса R і r, його висоту Н та довжину твірної l обрати довільно**.

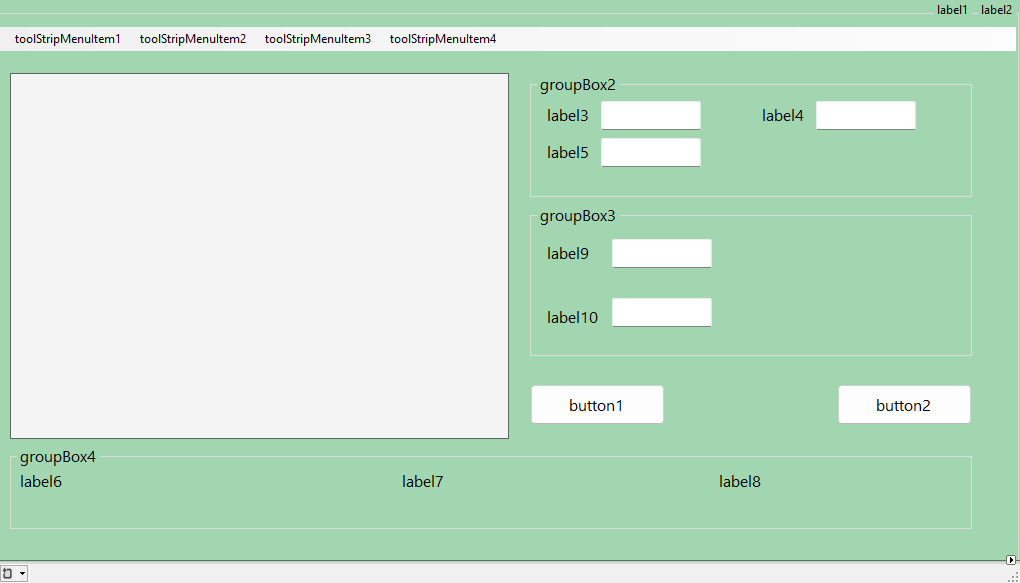


Рисунок 12 - Проектування форми для першого завдання

1. За допомогою властивостей різних елементів створено кінцевий вигляд форми (див. Рисунок 13)

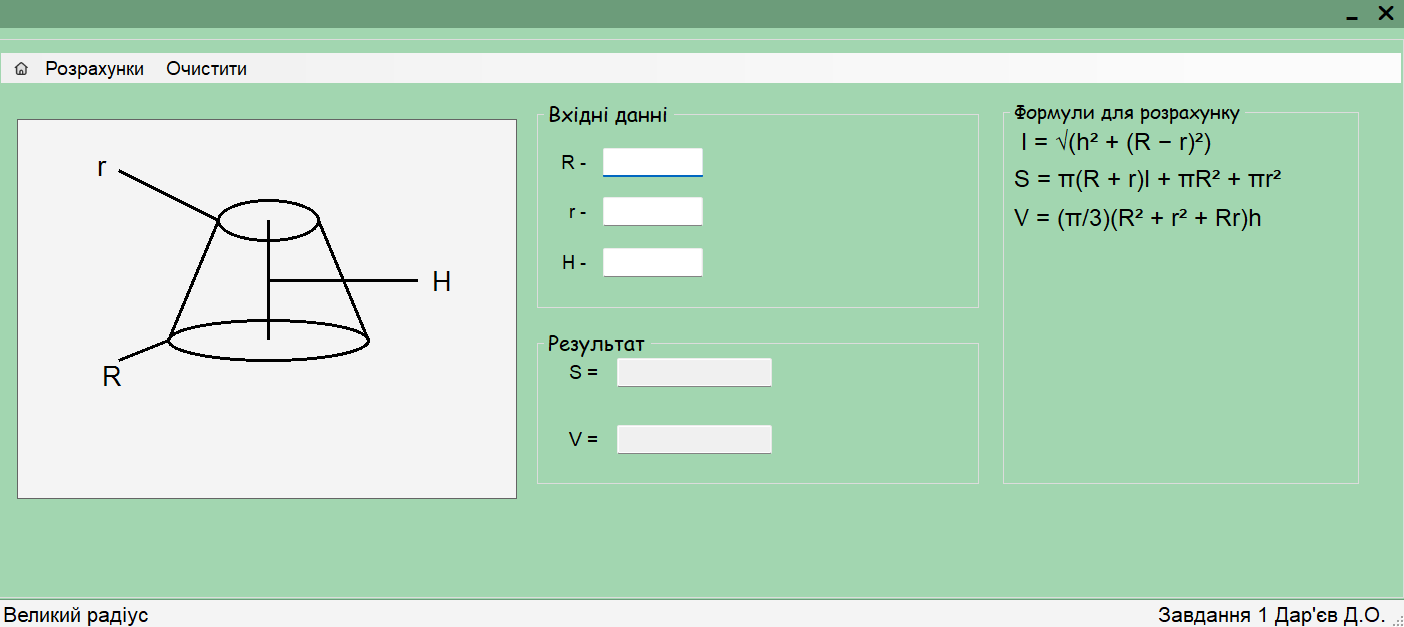


Рисунок 13 - Спроектована форма

1. Створено відповідний код для форми в просторі імен Індивідуальне завданняCSharp

#region Змінні для дизайна форми

private bool wasHiddenManually = false;

private bool dragging = false;

private Point dragCursorPoint;

private Point dragFormPoint;

#endregion

public TaskOneWindow()

{

InitializeComponent();

string relativePath = Path.Combine("Icons", "home-alt.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

toHomeStripMenuItem.Image = svgDoc.Draw(26, 26);

}

private void ControlInputData(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && e.KeyChar != '.' && !char.IsNumber(e.KeyChar) && e.KeyChar != '\b')

{

e.Handled = true;

}

if (e.KeyChar == '.' && (sender as TextBox).Text.Contains('.'))

{

e.Handled = true;

}

}

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

{

Graphics gr = e.Graphics;

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

Brush brush = new SolidBrush(Color.Black);

Rectangle rectangle = new Rectangle(200, 80, 100, 40);

gr.DrawEllipse(pen, rectangle);

rectangle = new Rectangle(150, 200, 200, 40);

gr.DrawEllipse(pen, rectangle);

gr.DrawLine(pen, new Point(200, 100), new Point(150, 220));

gr.DrawLine(pen, new Point(300, 100), new Point(350, 220));

gr.DrawLine(pen, new Point(250, 100), new Point(250, 220));

gr.DrawLine(pen, new Point(200, 100), new Point(100, 50));

gr.DrawString("r", new Font("Arial", 20), brush, new Point(75, 30));

gr.DrawLine(pen, new Point(150, 220), new Point(100, 240));

gr.DrawString("R", new Font("Arial", 20), brush, new Point(80, 240));

gr.DrawLine(pen, new Point(250, 160), new Point(400, 160));

gr.DrawString("H", new Font("Arial", 20), brush, new Point(410, 145));

}

private void taskOneTimer\_Tick(object sender, EventArgs e)

{

if (bigRadiusTextBox.Focused == true)

{

toolStripStatusLabel1.Text = "Великий радіус";

}

if (smallRadiusTextBox.Focused == true)

{

toolStripStatusLabel1.Text = "Малий радіус";

}

if (heightFigureTextBox.Focused == true)

{

toolStripStatusLabel1.Text = "Висота";

}

if (squareFigureTextBox.Focused == true)

{

toolStripStatusLabel1.Text = "Площа";

}

if (volumeFigureTextBox.Focused == true)

{

toolStripStatusLabel1.Text = "Об'єм";

}

}

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

{

bigRadiusTextBox.Text = null;

smallRadiusTextBox.Text = null;

heightFigureTextBox.Text = null;

squareFigureTextBox.Text = null;

volumeFigureTextBox.Text = null;

}

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

{

double figureSquare;

double figureVolume;

double generative;

double bigRadius = string.IsNullOrEmpty(bigRadiusTextBox.Text) ? double.NaN : Convert.ToDouble(bigRadiusTextBox.Text);

double smallRadius = string.IsNullOrEmpty(smallRadiusTextBox.Text) ? double.NaN : Convert.ToDouble(smallRadiusTextBox.Text);

double height = string.IsNullOrEmpty(heightFigureTextBox.Text) ? double.NaN : Convert.ToDouble(heightFigureTextBox.Text);

if (bigRadius == double.NaN || smallRadius == double.NaN || height == double.NaN)

{

MessageBox.Show("Введіть всі вхідні данні", "ERROR");

return;

}

if (bigRadius < smallRadius)

{

MessageBox.Show("Малий радіус більший ніж великий радіус (r > R)", "ERROR");

return;

}

generative = Math.Sqrt(height \* height + (bigRadius - smallRadius));

figureSquare = Math.PI \* (bigRadius + smallRadius) \* generative + Math.PI \* Math.Pow(bigRadius, 2) + Math.PI \* Math.Pow(smallRadius, 2);

figureVolume = (Math.PI / 3) \* (Math.Pow(bigRadius, 2) + Math.Pow(smallRadius, 2) + bigRadius \* smallRadius) \* height;

squareFigureTextBox.Text = string.Format("{0:F3}", figureSquare);

volumeFigureTextBox.Text = string.Format("{0:F3}", figureVolume);

}

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

{

Application.Exit();

}

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

{

wasHiddenManually = true;

this.WindowState = FormWindowState.Minimized;

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_RESTORE = 0xF120;

if (m.Msg == WM\_SYSCOMMAND && m.WParam.ToInt32() == SC\_RESTORE)

{

if (wasHiddenManually)

{

wasHiddenManually = false;

this.WindowState = FormWindowState.Normal;

Rectangle screen = Screen.FromControl(this).WorkingArea;

int x = (screen.Width - this.Width) / 2 + screen.Left;

int y = (screen.Height - this.Height) / 2 + screen.Top;

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

this.Opacity = 1;

this.Enabled = true;

}

}

base.WndProc(ref m);

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = true;

dragCursorPoint = Cursor.Position;

dragFormPoint = this.Location;

}

}

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

{

if (dragging)

{

Point diff = Point.Subtract(Cursor.Position, new Size(dragCursorPoint));

this.Location = Point.Add(dragFormPoint, new Size(diff));

}

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = false;

}

}

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

{

this.Hide();

MainMenuWindow mainMenu = new MainMenuWindow();

mainMenu.Show();

mainMenu.Activate();

}

1. Тестування роботи програми

Обчислення задачі згідно вхідних данних(див. Рисунок 13)

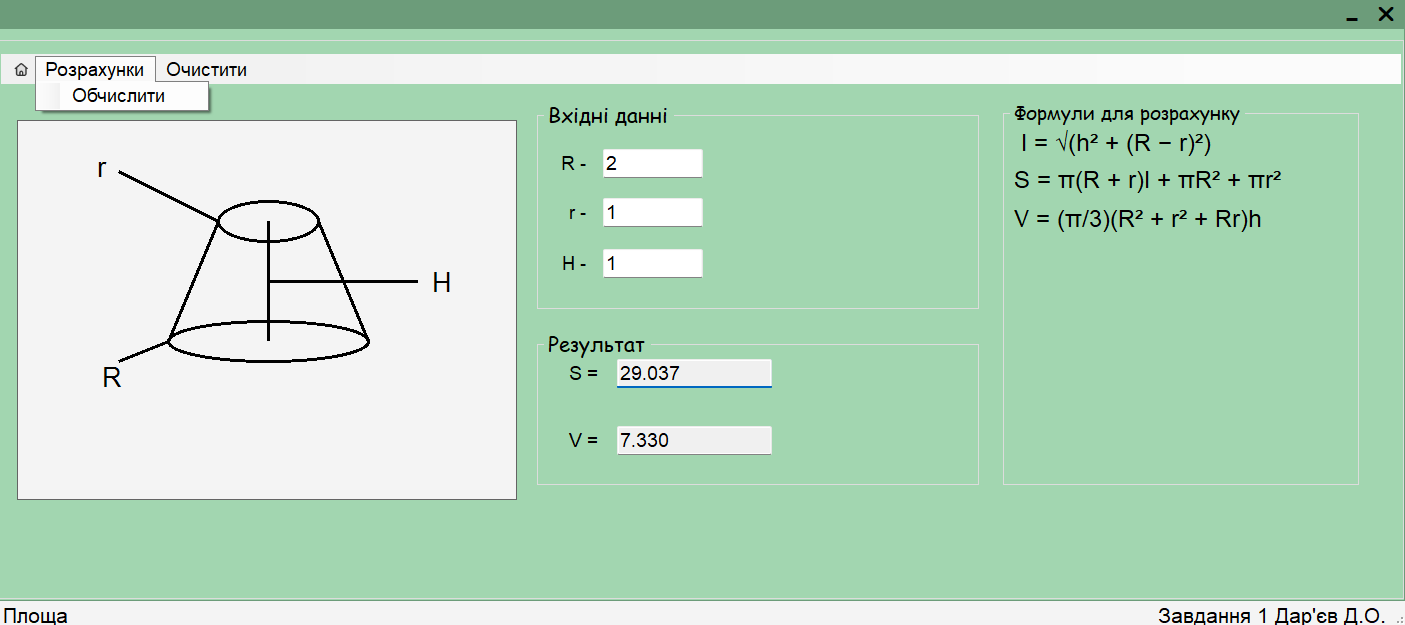


Рисунок 13 - Виконання розрахунків згідно завдання

Перевірка вхідних данних (див. Рисунок 14)

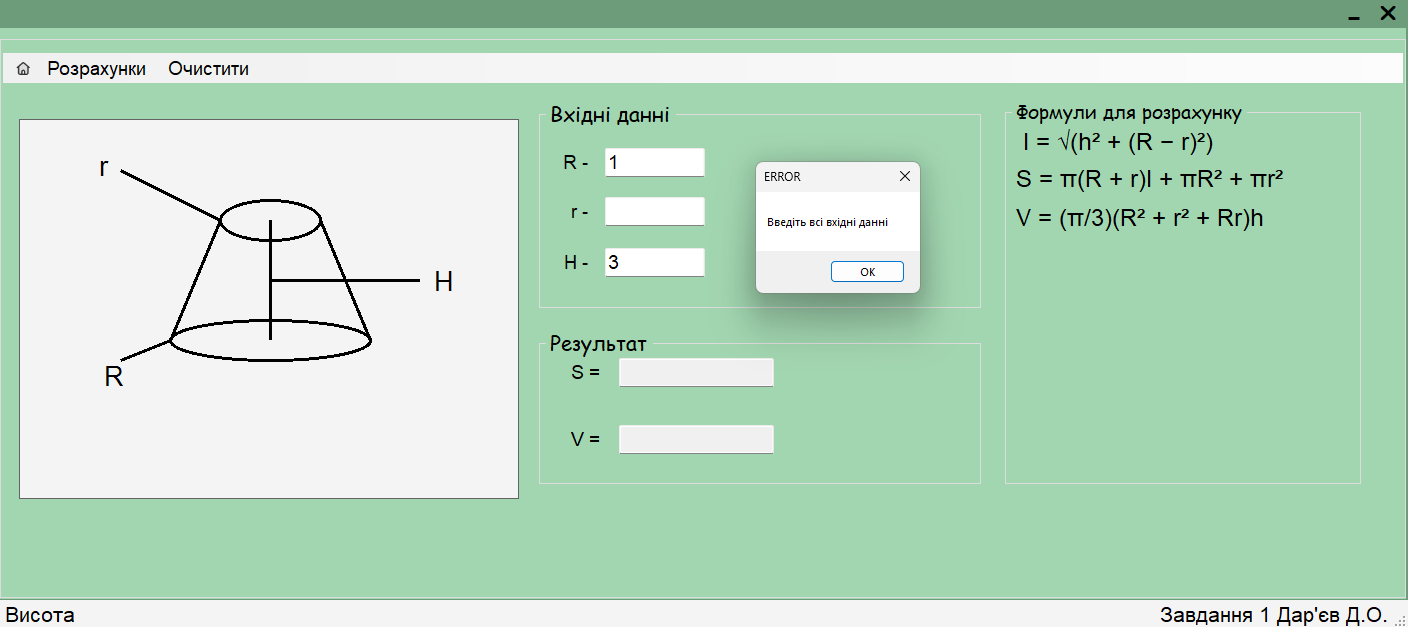


Рисунок 14 - Контроль вхідних данних

Завдання 2 Табуляція функції

1. Створимо ще одну форму з назвою TaskTwoWindow та додамо різні елементи керування (див. Рисунок 15) та виконати завдання:

**Виконати табуляцію функції та обчисли кількість і добуток 0,1<y<0,5. Забезпечити вивід значень х та у на екран, в масив та в текстовий файл, побудувати графік функції. Забезпечити читання файлу в проєкті.**

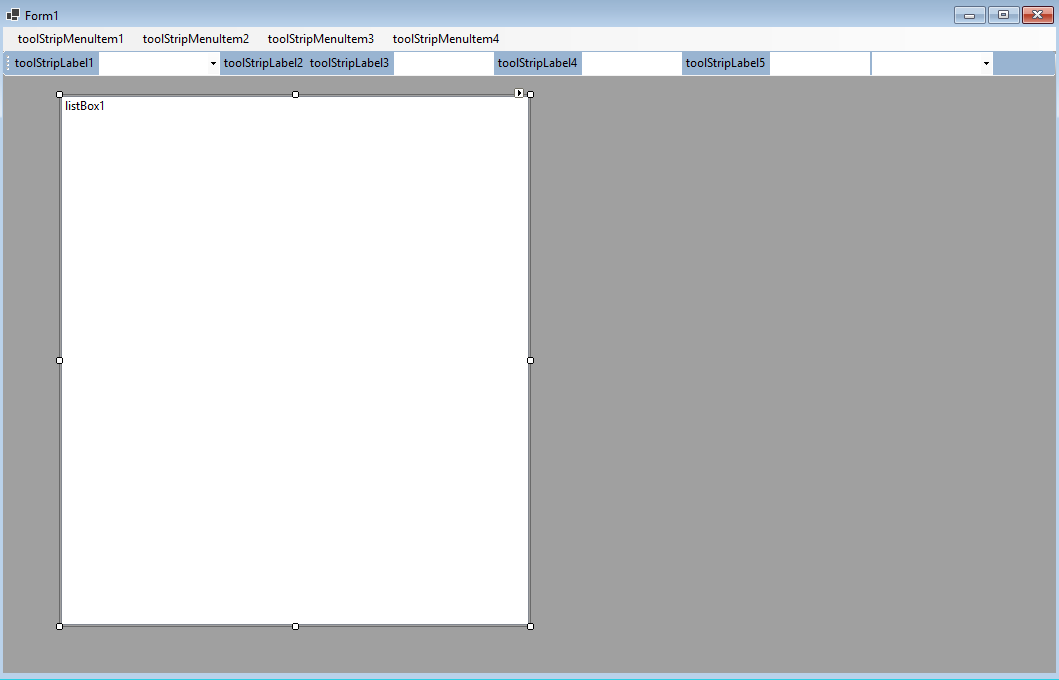


Рисунок 15 - Проектування форми для другого завдання

1. Заповнюємо форму та розміщуємо об’єкти для зручного користування(див. Рисунок 16).



Рисунок 16 - Кінцевий варіант форми

1. Створено програмний код для форми TaskTwoWindow в просторі імен ІндивідуальнеЗавданняCSharp

#region Змінні для дизайна форми

private bool wasHiddenManually = false;

private bool dragging = false;

private System.Drawing.Point dragCursorPoint;

private System.Drawing.Point dragFormPoint;

#endregion

private double[] resultArray;

public TaskTwoWindow()

{

InitializeComponent();

InitializeChart();

string relativePath = Path.Combine("Icons", "home-alt.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

timer1.Start();

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

toHomeStripMenuItem.Image = svgDoc.Draw(26, 26);

InstalledFontCollection fonts = new InstalledFontCollection();

foreach (FontFamily font in fonts.Families)

{

styleComboBox.Items.Add(font.Name);

}

superStyleComboBox.Items.Add("Regular");

superStyleComboBox.Items.Add("Bold");

superStyleComboBox.Items.Add("Italic");

superStyleComboBox.Items.Add("Underline");

superStyleComboBox.Items.Add("Strikeout");

styleComboBox.SelectedIndex = 0;

superStyleComboBox.SelectedIndex = 0;

fontSizeComboBox.SelectedIndex = 10;

}

private void InitializeChart()

{

oneChart.XAxes = new LiveChartsCore.SkiaSharpView.Axis[]

{

new LiveChartsCore.SkiaSharpView.Axis

{

Name = "x",

Labels = new List<string> { "1.1","1.2","1.3","1.4","1.5", "1.6", "1.7", "1.8", "1.9", "2.0" }

}

};

oneChart.YAxes = new LiveChartsCore.SkiaSharpView.Axis[]

{

new LiveChartsCore.SkiaSharpView.Axis

{

Name = "y",

Labels = new List<string> { "0.0", "0.01", "0.02", "0.03", "0.04", "0.05", "0.06", "0.07", "0.08", "0.09", "0.10", "0.11",

"0.12", "0.13", "0.14", "0.15", "0.16", "0.17", "0.18", "0.19", "0.20" },

}

};

oneChart.ZoomMode = LiveChartsCore.Measure.ZoomAndPanMode.X;

oneChart.Update();

}

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

{

Application.Exit();

}

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

{

wasHiddenManually = true;

this.WindowState = FormWindowState.Minimized;

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_RESTORE = 0xF120;

if (m.Msg == WM\_SYSCOMMAND && m.WParam.ToInt32() == SC\_RESTORE)

{

if (wasHiddenManually)

{

wasHiddenManually = false;

this.WindowState = FormWindowState.Normal;

Rectangle screen = Screen.FromControl(this).WorkingArea;

int x = (screen.Width - this.Width) / 2 + screen.Left;

int y = (screen.Height - this.Height) / 2 + screen.Top;

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

this.Opacity = 1;

this.Enabled = true;

}

}

base.WndProc(ref m);

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = true;

dragCursorPoint = System.Windows.Forms.Cursor.Position;

dragFormPoint = this.Location;

}

}

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

{

if (dragging)

{

System.Drawing.Point diff = System.Drawing.Point.Subtract(System.Windows.Forms.Cursor.Position, new Size(dragCursorPoint));

this.Location = System.Drawing.Point.Add(dragFormPoint, new Size(diff));

}

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = false;

}

}

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

{

this.Hide();

MainMenuWindow mainMenu = new MainMenuWindow();

mainMenu.Show();

mainMenu.Activate();

}

private void DataControlInput(object sender, KeyPressEventArgs e)

{

ToolStripTextBox tb = sender as ToolStripTextBox;

if (tb == null)

return;

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

{

e.Handled = true;

return;

}

if (e.KeyChar == '.' && tb.Text.Contains('.'))

{

e.Handled = true;

return;

}

if (e.KeyChar == '-')

{

if (tb.Text.Contains('-') || tb.SelectionStart != 0)

{

e.Handled = true;

return;

}

}

}

private double ExpressionOne(double x)

{

double result;

if (x <= 0)

throw new Exception("Значення x повинно бути більше 0");

result = Math.Log(x) / (x \* x \* x);

return result;

}

private double ProdAndCountYBetween(double left, double right, double step, out double countY)

{

if (left > right && step > 0)

throw new Exception("Ліва межа ніколи не добереться до правої (ліва межа більша за праву і прямує до нескіченності)");

if (left < right && step < 0)

throw new Exception("Ліва межа ніколи не добереться до правої (ліва межа менша за праву і прямує до відємної нескіченності)");

double prodY = 1;

countY = 0;

if (left < right && step > 0)

{

for (double i = left; i < right; i += Math.Abs(step))

{

double y = ExpressionOne(i);

if (y > 0.2 && y < 0.5)

{

prodY \*= y;

countY++;

}

}

}

if (left > right && step < 0)

{

for (double i = left; i > right; i -= Math.Abs(step))

{

double y = ExpressionOne(i);

if (y > 0.2 && y < 0.5)

{

prodY \*= y;

countY++;

}

}

}

if (left == right || step == 0)

{

double y = ExpressionOne(left);

if (y > 0.2 && y < 0.5)

{

prodY \*= y;

countY++;

}

}

return prodY;

}

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

{

if (!double.TryParse(borderLeftToolStripTextBox.Text, out double left) ||

!double.TryParse(borderRightToolStripTextBox.Text, out double right) ||

!double.TryParse(stepToolStripTextBox.Text, out double step))

{

MessageBox.Show("Введіть всі числові значення в усі поля", "ERROR");

return;

}

List<(double X, double Y)> values;

double countY;

double prodY;

try

{

prodY = ProdAndCountYBetween(left, right, step, out countY);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ERROR");

return;

}

try

{

values = GenerateData(left, right, step);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ERROR");

return;

}

if (outInToFileToolStripMenuItem.Checked)

{

string path = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "TaskTwoFile.txt");

using StreamWriter sw = new StreamWriter(path);

sw.WriteLine($"{"X",-8}{"Y",8}");

foreach (var (x, y) in values)

sw.WriteLine($"{x,-8:F5} {y,8:F5}");

sw.WriteLine($"Кількість Y: {countY}");

sw.WriteLine($"А добуток = {prodY}");

}

if (outInToArrayToolStripMenuItem.Checked)

{

resultArray = new double[values.Count \* 2];

int i = 0;

foreach (var (x, y) in values)

{

resultArray[i++] = x;

resultArray[i++] = y;

}

}

if (outOnToScreenToolStripMenuItem.Checked)

{

listBox.Items.Add($"{"X",-8}{"Y",8}");

foreach (var (x, y) in values)

listBox.Items.Add($"{x,-8:F3} {y,-14:F5}");

listBox.Items.Add($"Кількість Y: {countY}");

listBox.Items.Add($"А добуток = {prodY}");

}

}

private List<(double X, double Y)> GenerateData(double left, double right, double step)

{

if (step == 0)

throw new ArgumentException("Крок не може дорівнювати 0");

var result = new List<(double X, double Y)>();

if (left < right && step > 0)

{

for (double i = left; i < right; i += step)

result.Add((i, ExpressionOne(i)));

}

else if (left > right && step < 0)

{

for (double i = left; i > right; i += step)

result.Add((i, ExpressionOne(i)));

}

else

{

throw new ArgumentException("Невірне поєднання меж і кроку.");

}

return result;

}

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

{

if ((sender as ToolStripMenuItem).Checked == true)

{

(sender as ToolStripMenuItem).Checked = false;

}

else (sender as ToolStripMenuItem).Checked = true;

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (outInToArrayToolStripMenuItem.Checked == true)

{

arrayOnScreenToolStripButton.Visible = true;

}

else arrayOnScreenToolStripButton.Visible = false;

}

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

{

listBox.Items.Clear();

borderLeftToolStripTextBox.Clear();

borderRightToolStripTextBox.Clear();

stepToolStripTextBox.Clear();

}

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

{

if (resultArray == null || resultArray.Length == 0)

{

MessageBox.Show("Масив порожній або не ініціалізований", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (resultArray.Length % 2 != 0)

{

MessageBox.Show("Дані в масиві некоректні (очікується парна кількість значень)", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

listBox.Items.Add($"{"X",-8}{"Y",8}");

for (int i = 0; i < resultArray.Length; i += 2)

{

listBox.Items.Add($"{resultArray[i],-8:F3} {resultArray[i + 1],-14:F5}");

}

}

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

{

ColorDialog fontDialog = new ColorDialog();

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

{

listBox.ForeColor = fontDialog.Color;

fontColorTextBox.BackColor = fontDialog.Color;

}

}

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

{

styleToolStrip.Visible = true;

calculateToolStrip.Visible = false;

}

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

{

styleToolStrip.Visible = false;

calculateToolStrip.Visible = true;

}

private System.Drawing.FontStyle GetFontStyle(string styleName)

{

switch (styleName)

{

case "Regular":

return System.Drawing.FontStyle.Regular;

case "Bold":

return System.Drawing.FontStyle.Bold;

case "Italic":

return System.Drawing.FontStyle.Italic;

case "Underline":

return System.Drawing.FontStyle.Underline;

case "Strikeout":

return System.Drawing.FontStyle.Strikeout;

default:

return System.Drawing.FontStyle.Regular;

}

}

private void ApplyFont()

{

if (styleComboBox.SelectedItem == null || superStyleComboBox.SelectedItem == null || fontSizeComboBox.SelectedItem == null)

{

styleComboBox.SelectedIndex = 0;

superStyleComboBox.SelectedIndex = 0;

fontSizeComboBox.SelectedIndex = 10;

return;

}

string? selectedFont = styleComboBox.SelectedItem as string;

string? selectedStyle = superStyleComboBox.SelectedItem as string;

if (string.IsNullOrEmpty(selectedFont) || string.IsNullOrEmpty(selectedStyle))

{

MessageBox.Show("Будь ласка, виберіть шрифт і стиль.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (!float.TryParse(fontSizeComboBox.SelectedItem.ToString(), out float fontSize))

{

MessageBox.Show("Вибрано недійсний розмір шрифту.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

System.Drawing.FontStyle style = GetFontStyle(selectedStyle);

listBox.Font = new Font(selectedFont, fontSize, style);

}

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

{

ApplyFont();

}

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

{

ApplyFont();

}

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

{

ApplyFont();

}

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

{

if (!double.TryParse(borderLeftToolStripTextBox.Text, out double left) ||

!double.TryParse(borderRightToolStripTextBox.Text, out double right) ||

!double.TryParse(stepToolStripTextBox.Text, out double step))

{

MessageBox.Show("Введіть всі числові значення в усі поля", "ERROR");

return;

}

List<(double X, double Y)> values;

try

{

values = GenerateData(left, right, step);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "ERROR");

return;

}

double[] valueX = new double[values.Count];

double[] valueY = new double[values.Count];

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

{

valueX[i] = values[i].X;

valueY[i] = values[i].Y;

}

oneChart.Series = new ISeries[]

{

new LineSeries<double>

{

Name = "Точка",

Values = valueY,

Fill = null,

GeometrySize = 5,

Stroke = new SolidColorPaint(SKColors.Red) { StrokeThickness = 1 }

}

};

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Зображення|\*.jpg";

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

{

try

{

using (var bitmap = new Bitmap(oneChart.Width, oneChart.Height))

{

oneChart.DrawToBitmap(bitmap, new Rectangle(0, 0, bitmap.Width, bitmap.Height));

bitmap.Save(saveFileDialog.FileName, System.Drawing.Imaging.ImageFormat.Jpeg);

}

MessageBox.Show("Зображення збережено успішно!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при збереженні зображення: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Текстовий файл|\*.txt";

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

{

try

{

using (StreamReader reader = new StreamReader(openFileDialog.FileName))

{

string line;

listBox.Items.Clear();

while ((line = reader.ReadLine()) != null)

{

listBox.Items.Add(line);

}

}

MessageBox.Show("Файл відкрито успішно!", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при відкритті файлу: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

1. Тестування функцій форми TaskTwoWindow

Табулювання функції та виведення її на графік

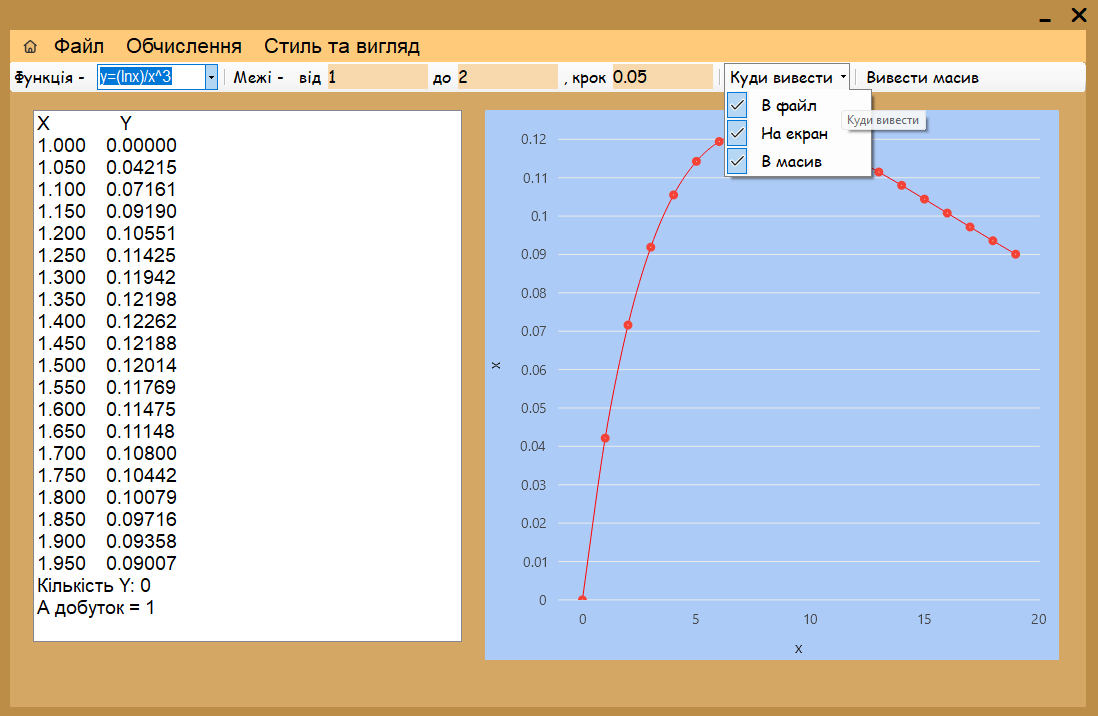


Рисунок 17 - Табулювання та вивід функції

Зміна кольору,стилю,шрифту,розміру тексту в виводі на екран(див. Рисунок 18)

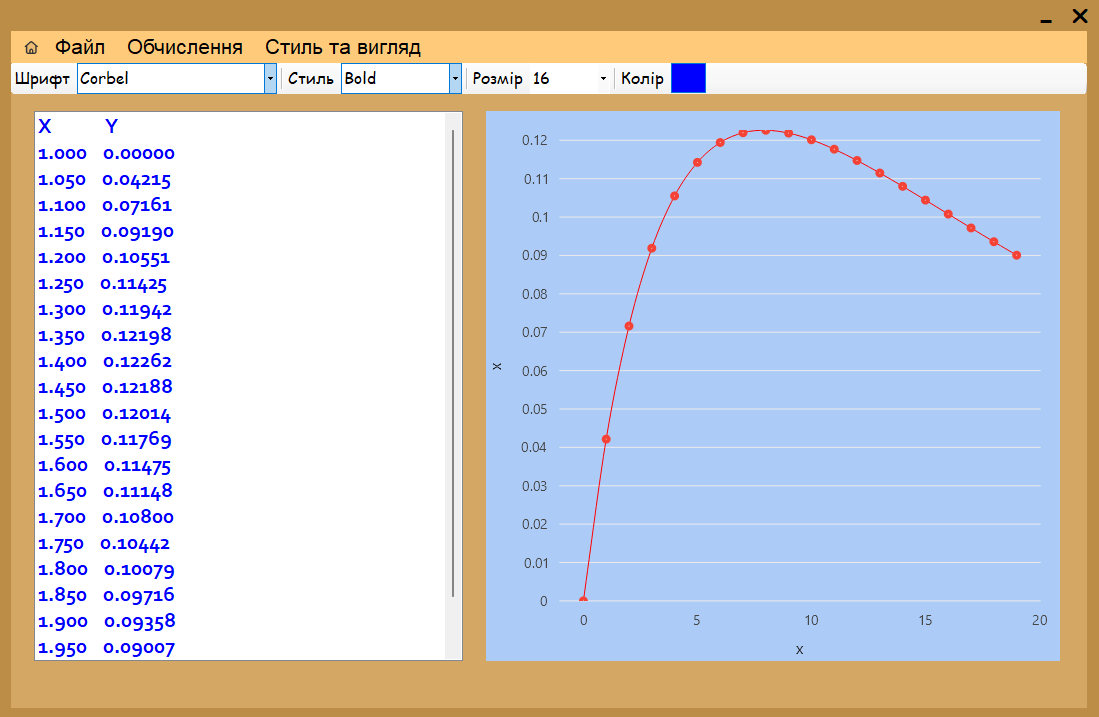


Рисунок 18 - Форматування тексту

Зберігання взображення графіка файл на пристрої(див. Рисунок 19)

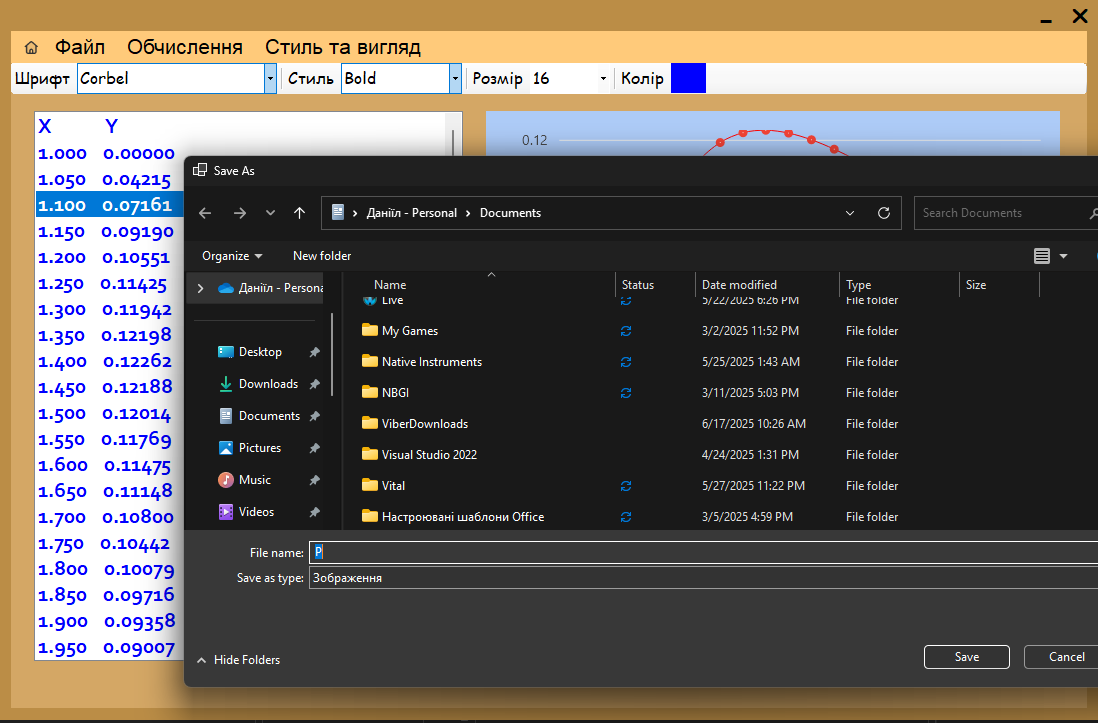


Рисунок 19 - Зберігання зображення

Перевірка при введені невірних вхідних данних(див. Рисунок 20/21)

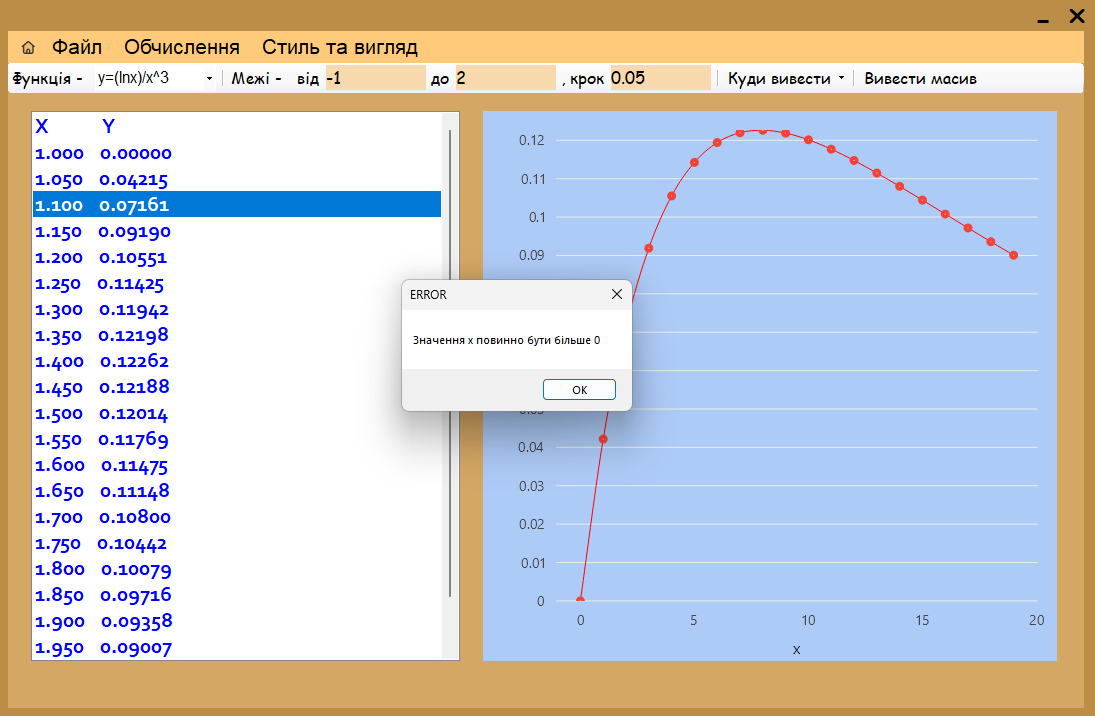


Рисунок 20 - Невірно введено межу

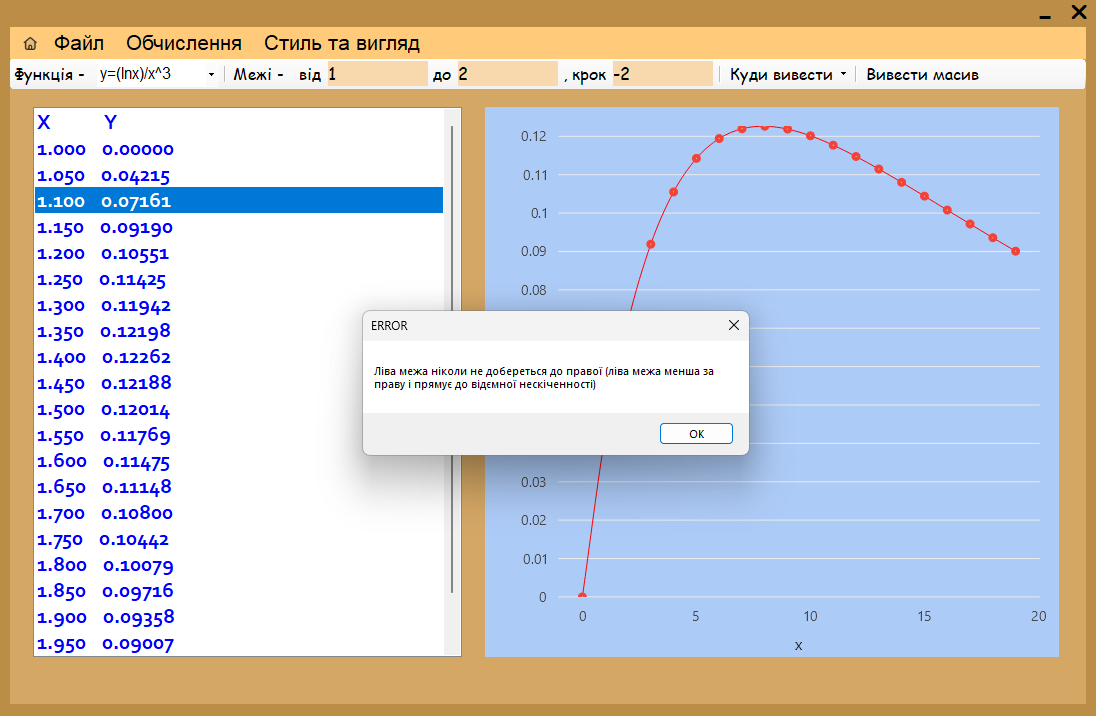


Рисунок 21 - Невірно введено крок

Завдання 3 Виконання дій над масивом

1. Створимо форму з назвою ThirdTaskWindow в просторі імен ІндивідуальнеЗавданняCSharp та заповнимо її елементами(див. Рисунок 22) для виконання завдання

**У одновимірному масиві, що складається з n дійсних елементів, обчислити:**

**1) максимальний елемент масиву;**

**2) суму елементів масиву, розташованих до останнього позитивного елементу.**

**Стиснути масив, видаливши з нього всі елементи, модуль яких знаходиться в інтервалі [а, Ь]. Елементи, що звільнилися в кінці масиву, заповнити нулями.**

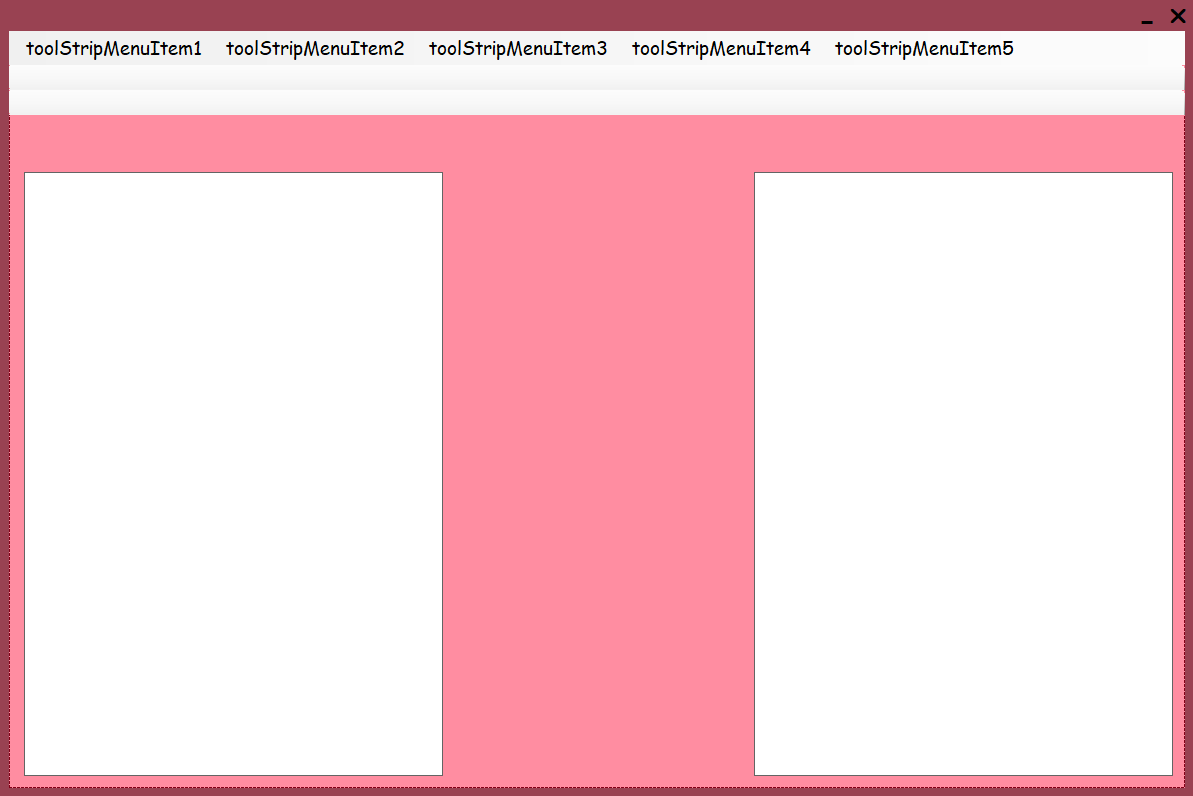


Рисунок 22 - Розробка дизайну для третьої задачі

1. За допомогою Інспектора об’єктів та Властивостей об’єктів розташовуємо та форматуємо форму(див. Рисунок 23)

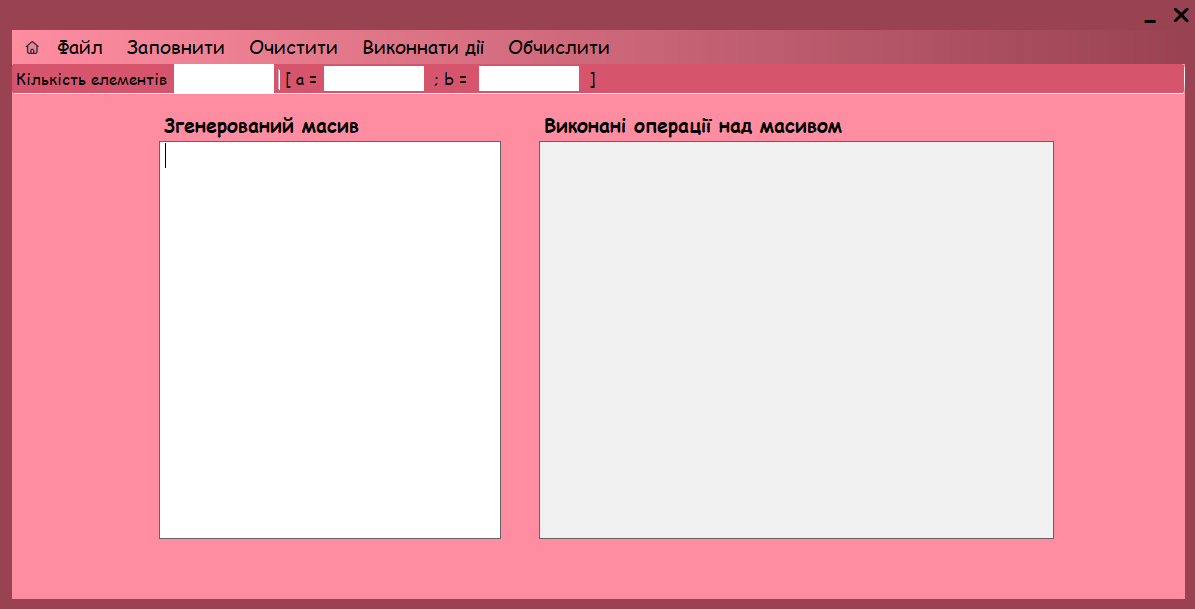


Рисунок 23 - Кінцевий варіант форми

1. Написано відповідний код до форми TaskThreeWindow в просторі імен ІндивідуальнеЗавданняCSharp

#region Змінні для дизайна форми

private bool wasHiddenManually = false;

private bool dragging = false;

private System.Drawing.Point dragCursorPoint;

private System.Drawing.Point dragFormPoint;

private Color colorOne;

private Color colorTwo;

#endregion

#region Змінні для завдання

private double maxValue;

private double valueSumToLastPosEl;

private double[] generatedArray;

private double[] filteredArray;

#endregion

public TaskThreeWindow()

{

InitializeComponent();

string relativePath = Path.Combine("Icons", "home-alt.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

toHomeStripMenuItem.Image = svgDoc.Draw(26, 26);

colorOne = panel1.BackColor;

colorTwo = this.BackColor;

menuStrip1.Refresh();

}

#region Обробка подій для розташування форми

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

{

Application.Exit();

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_RESTORE = 0xF120;

if (m.Msg == WM\_SYSCOMMAND && m.WParam.ToInt32() == SC\_RESTORE)

{

if (wasHiddenManually)

{

wasHiddenManually = false;

this.WindowState = FormWindowState.Normal;

Rectangle screen = Screen.FromControl(this).WorkingArea;

int x = (screen.Width - this.Width) / 2 + screen.Left;

int y = (screen.Height - this.Height) / 2 + screen.Top;

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

this.Opacity = 1;

this.Enabled = true;

}

}

base.WndProc(ref m);

}

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

{

wasHiddenManually = true;

this.WindowState = FormWindowState.Minimized;

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = true;

dragCursorPoint = System.Windows.Forms.Cursor.Position;

dragFormPoint = this.Location;

}

}

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

{

if (dragging)

{

System.Drawing.Point diff = System.Drawing.Point.Subtract(System.Windows.Forms.Cursor.Position, new Size(dragCursorPoint));

this.Location = System.Drawing.Point.Add(dragFormPoint, new Size(diff));

}

}

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

{

if (e.Button == MouseButtons.Left)

{

dragging = false;

}

}

#endregion

private void menuStrip1\_Paint(object sender, PaintEventArgs e)

{

Rectangle rect = new Rectangle(menuStrip1.Location.X, menuStrip1.Location.Y, menuStrip1.Width, menuStrip1.Height);

LinearGradientBrush gradientBrush = new LinearGradientBrush(

rect,

colorOne,

colorTwo,

LinearGradientMode.Horizontal);

e.Graphics.FillRectangle(gradientBrush, rect);

}

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

{

this.Hide();

MainMenuWindow mainMenu = new MainMenuWindow();

mainMenu.Show();

mainMenu.Activate();

}

private void toolStripMenuItem3\_Click(object sender, EventArgs e)

{

generatedArrayTextBox.Clear();

endResultTextBox.Clear();

}

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

{

if (!char.IsNumber(e.KeyChar) && !char.IsDigit(e.KeyChar) && e.KeyChar != '\b')

{

e.Handled = true;

}

}

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

{

int size;

Random rnd = new Random();

if (!int.TryParse(sizeArrayToolStripTextBox.Text, out size))

{

MessageBox.Show("Помилка конвертації", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

generatedArray = new double[size];

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

generatedArray[i] = rnd.Next() % 1001 - 500;

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

{

generatedArrayTextBox.Text = generatedArrayTextBox.Text + string.Format("{0:F2}, ", generatedArray[i]);

}

}

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

{

if ((sender as ToolStripMenuItem).CheckState == CheckState.Checked)

{

(sender as ToolStripMenuItem).CheckState = CheckState.Unchecked;

}

else (sender as ToolStripMenuItem).CheckState = CheckState.Checked;

}

private double[] ConverteTextBoxToArray()

{

string input = generatedArrayTextBox.Text;

if (string.IsNullOrWhiteSpace(input))

{

throw new ArgumentNullException("Введіть значення масиву у текстове поле.");

}

input = input.Replace(",", " ");

input = System.Text.RegularExpressions.Regex.Replace(input, @"\s+", " ").Trim();

string[] parts = input.Split(new char[] { ' ' }, StringSplitOptions.RemoveEmptyEntries);

List<double> numbers = new List<double>();

List<string> errors = new List<string>();

foreach (string part in parts)

{

if (double.TryParse(part, out double num))

{

numbers.Add(num);

}

else

{

errors.Add(part);

}

}

if (errors.Count > 0)

{

MessageBox.Show("Некоректні значення: " + string.Join(", ", errors));

}

return numbers.ToArray();

}

private double calculateMaxValue(double[] generatedArray)

{

try

{

maxValue = generatedArray.Max();

return maxValue;

}

catch (ArgumentNullException ex)

{

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

}

return maxValue;

}

private double calculateSumToLastPositiveElement(double[] generatedArray)

{

double sum = 0;

int lastPositiveIndex = -1;

try

{

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

{

if (generatedArray[i] >= 0)

lastPositiveIndex = i;

}

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

sum += generatedArray[i];

return sum;

}

catch (ArgumentNullException ex)

{

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

}

return sum;

}

private double[] compressArray(double[] generatedArray, double a, double b)

{

double[] resultArray = new double[generatedArray.Length];

if (a > b) {

double temp;

temp = a;

a = b;

b = temp;

}

for (int j = 0, i = 0; i < generatedArray.Length; i++)

{

if (Math.Abs(generatedArray[i]) <= a || Math.Abs(generatedArray[i]) >= b)

{

resultArray[j] = generatedArray[i];

j++;

}

}

return resultArray;

}

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

{

generatedArray = ConverteTextBoxToArray();

if (maxElemToolStripMenuItem.CheckState == CheckState.Checked)

{

endResultTextBox.Text += ("Максимальне значення: " + calculateMaxValue(generatedArray).ToString("F2") + Environment.NewLine);

}

if (sumToLastPosToolStripMenuItem.CheckState == CheckState.Checked)

{

endResultTextBox.Text += ("Сума до останього позитивного: " + calculateSumToLastPositiveElement(generatedArray).ToString("F2") + Environment.NewLine);

}

if (compressChekToolStripMenuItem.CheckState == CheckState.Checked)

{

double a, b;

if (!double.TryParse(AAAToolStripTextBox.Text, out a) || !double.TryParse(BBBToolStripTextBox.Text, out b))

{

MessageBox.Show("Помилка: Введіть коректні значення для A та B.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

double[] compressedArray = compressArray(generatedArray, a, b);

endResultTextBox.Text += ("Стиснутий масив: ");

foreach (var compressed in compressedArray)

{

endResultTextBox.Text += compressed.ToString("F2") + ", ";

}

endResultTextBox.Text = endResultTextBox.Text.TrimEnd(',', ' ') + Environment.NewLine;

}

}

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

{

if (!char.IsDigit(e.KeyChar) && !char.IsNumber(e.KeyChar) && e.KeyChar != ' ' && e.KeyChar != '-' && e.KeyChar != '\b' && e.KeyChar != '.')

e.Handled = true;

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (compressChekToolStripMenuItem.CheckState == CheckState.Checked)

{

AAAToolStripTextBox.Visible = true;

BBBToolStripTextBox.Visible = true;

toolStripLabel2.Visible = true;

toolStripLabel3.Visible = true;

toolStripLabel4.Visible = true;

}

else

{

AAAToolStripTextBox.Visible = false;

BBBToolStripTextBox.Visible = false;

toolStripLabel2.Visible = false;

toolStripLabel3.Visible = false;

toolStripLabel4.Visible = false;

}

}

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

{

if (!char.IsDigit(e.KeyChar) && !char.IsNumber(e.KeyChar) && e.KeyChar != '\b' && e.KeyChar != '.')

{

e.Handled = true;

}

if (e.KeyChar == '.' && (sender as ToolStripTextBox).Text.Contains("."))

{

e.Handled = true;

}

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Text files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*",

Title = "Зберегти результати"

};

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

{

try

{

using (StreamWriter writer = new StreamWriter(saveFileDialog.FileName))

{

writer.WriteLine("Згенерований масив:");

writer.WriteLine(generatedArrayTextBox.Text);

writer.WriteLine();

writer.WriteLine("Результати обчислень:");

writer.WriteLine(endResultTextBox.Text);

}

MessageBox.Show("Файл успішно збережено.", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

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

}

}

}

private void toolStripMenuItem4\_Click(object sender, EventArgs e)

{

endResultTextBox.Text = null;

}

private void toolStripMenuItem2\_Click(object sender, EventArgs e)

{

generatedArrayTextBox.Text = null;

}

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

sizeArrayToolStripTextBox.Text = null;

AAAToolStripTextBox.Text = null;

BBBToolStripTextBox.Text = null;

}

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

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Text files (\*.txt)|\*.txt",

Title = "Відкрити файл"

};

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

{

string filePath = openFileDialog.FileName;

try

{

using (StreamReader reader = new StreamReader(filePath)) {

string fileContent = reader.ReadToEnd();

generatedArrayTextBox.Text = fileContent;

}

}

catch (Exception ex)

{

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

}

}

}

1. Протестовану роботу програми при різних вхідних данних та введення з файлу, власноруч, та згенеровані випадково. Результат роботи збережено в файл(див Рисунок 24/25/26/27/28).

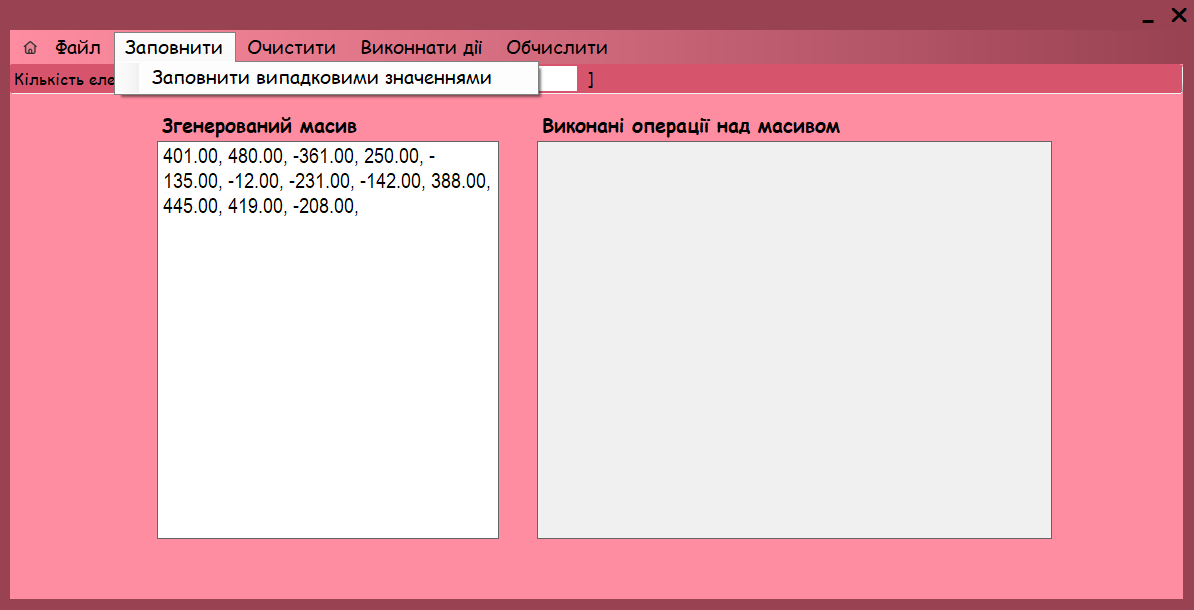


Рисунок 24 - Заповнення масиву випадковими значеннями

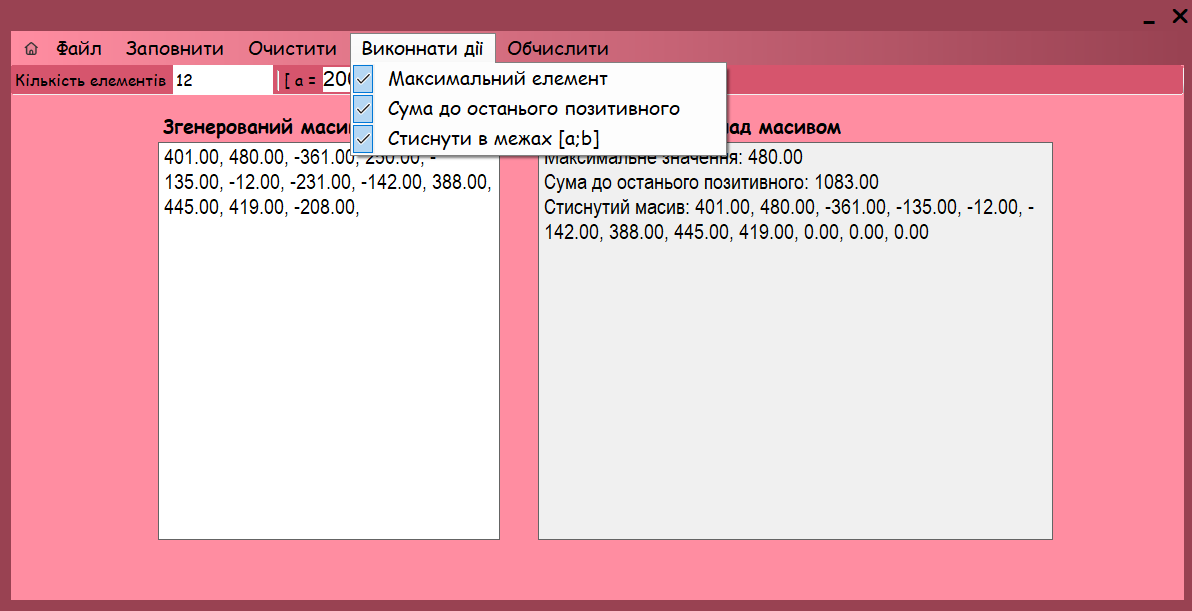


Рисунок 25 - Виконання дій згідно заданої умови

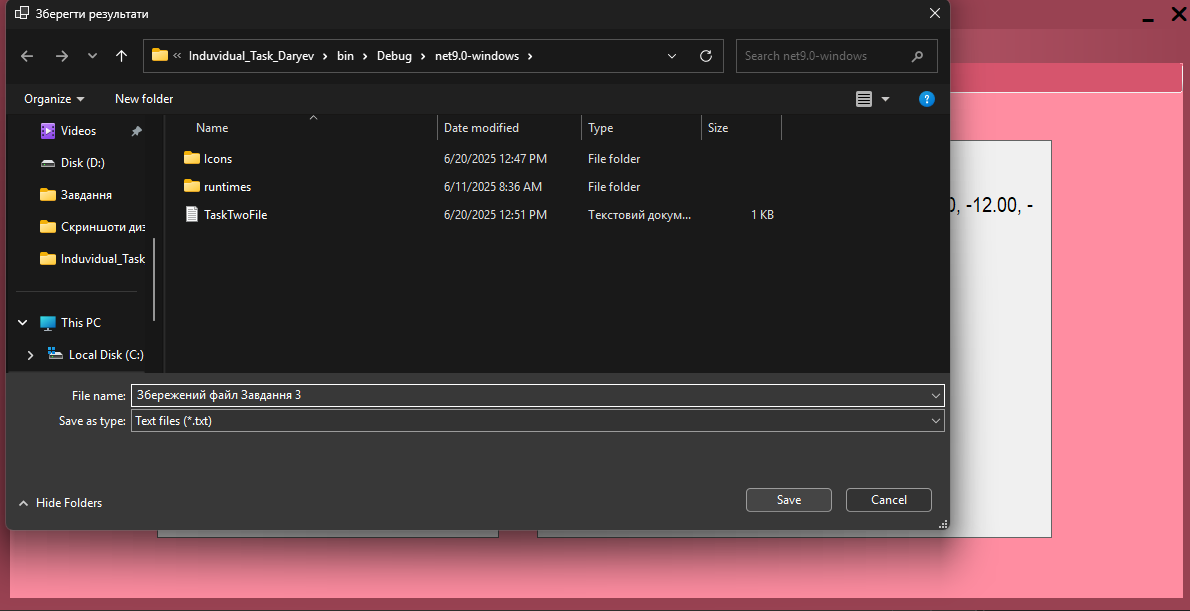


Рисунок 26 - Збереження файлу на диску

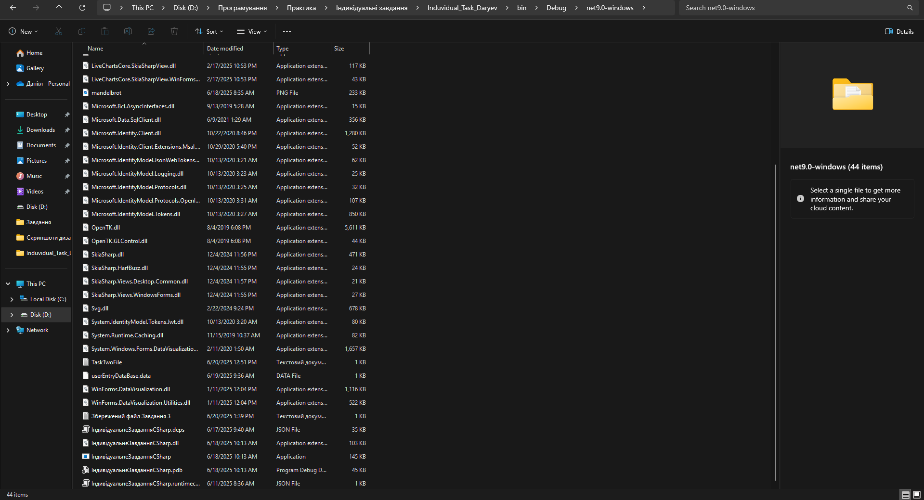


Рисунок 27 - Збережений файл на диску

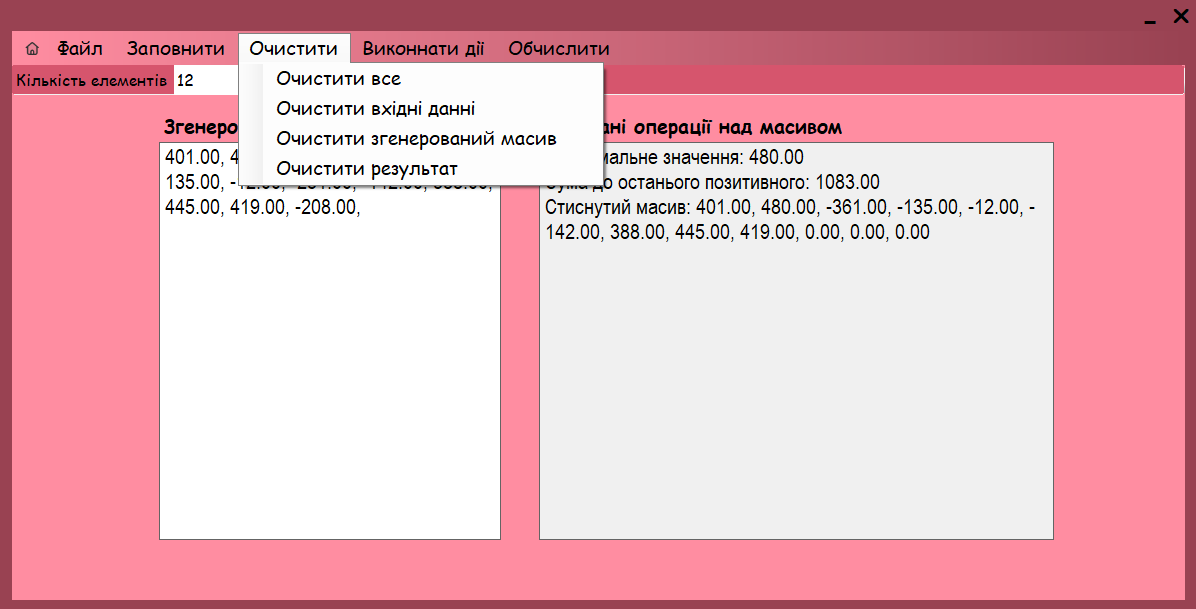


Рисунок 28 - Вкладка Очистити

Завдання 4 Побудова графічних зображень на основі математичних функцій.

1. Створюємо нову форму з назвою TaskFourWindow в просторі імен ІндивідуальнеЗавданняCSharp та розміщуємо всі необхідні об’єкти(див. Рисунок 29)

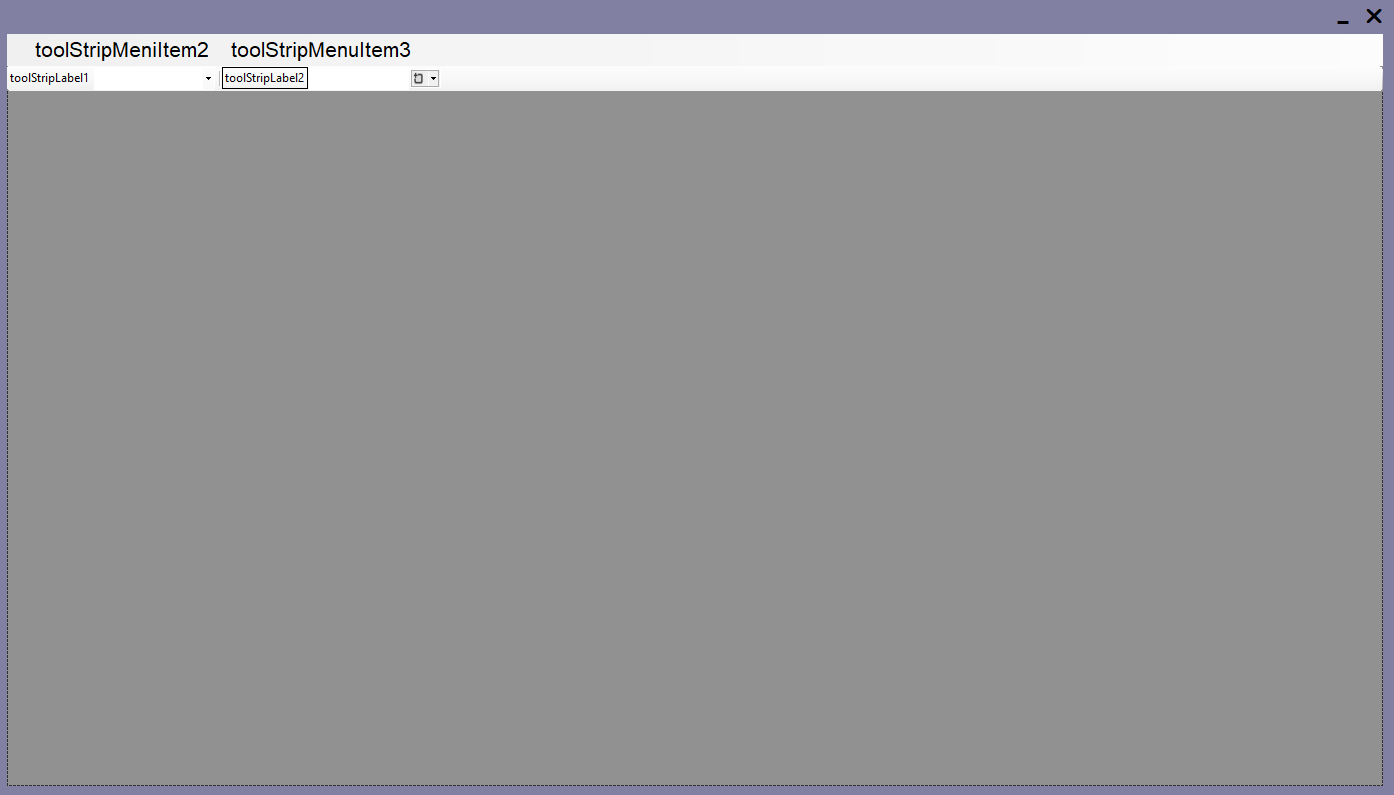


Рисунок 29 - Проектування форми TaskFourWindow

1. За допомогою засобів форматування змінюємо вигляд та вміст об’єктів(див. Рисунок 30)

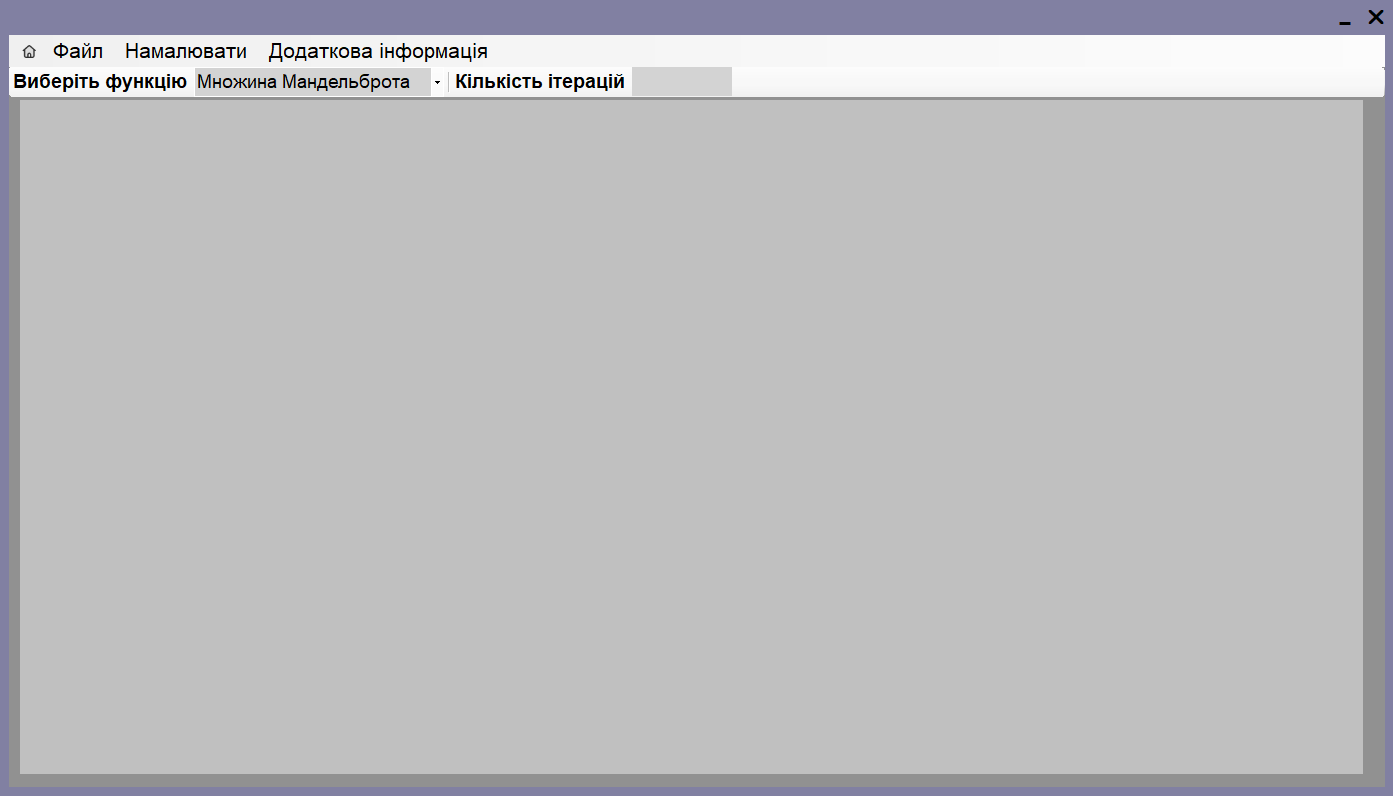


Рисунок 30 - Кінцевий варіант форми

1. Написано відповідний програмний код до форми

#region Змінні для дизайну форми

private const int WM\_NCLBUTTONDOWN = 0xA1;

private const int HT\_CAPTION = 0x2;

[DllImport("user32.dll")]

public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

[DllImport("user32.dll")]

public static extern bool ReleaseCapture();

private const int HT\_LEFT = 10;

private const int HT\_RIGHT = 11;

private const int HT\_TOP = 12;

private const int HT\_TOPLEFT = 13;

private const int HT\_TOPRIGHT = 14;

private const int HT\_BOTTOM = 15;

private const int HT\_BOTTOMLEFT = 16;

private const int HT\_BOTTOMRIGHT = 17;

private const int BorderSize = 8; // Ширина зони, за яку можна "тягнути" форму

private bool wasHiddenManually = false;

#endregion

private double centerX = -0.5;

private double centerY = 0;

private double scale = 3.5;

private int maxIterations = 100;

public TaskFourWindow()

{

InitializeComponent();

string relativePath = Path.Combine("Icons", "home-alt.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

toHomeStripMenuItem.Image = svgDoc.Draw(26, 26);

this.DoubleBuffered = true;

this.SetStyle(ControlStyles.ResizeRedraw, true);

pictureBox1.MouseEnter += (s, e) => pictureBox1.Focus();

pictureBox1.MouseWheel += PictureBox1\_MouseWheel;

toolStripComboBox1.SelectedItem = "Множина Мандельброта";

}

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_RESTORE = 0xF120;

if (m.Msg == WM\_SYSCOMMAND && m.WParam.ToInt32() == SC\_RESTORE)

{

if (wasHiddenManually)

{

wasHiddenManually = false;

this.WindowState = FormWindowState.Normal;

Rectangle screen = Screen.FromControl(this).WorkingArea;

int x = (screen.Width - this.Width) / 2 + screen.Left;

int y = (screen.Height - this.Height) / 2 + screen.Top;

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

this.Opacity = 1;

this.Enabled = true;

}

base.WndProc(ref m);

return;

}

base.WndProc(ref m);

if (m.Msg == 0x84) // WM\_NCHITTEST

{

Point pos = new Point(m.LParam.ToInt32());

pos = this.PointToClient(pos);

if (pos.X <= BorderSize && pos.Y <= BorderSize) { m.Result = (IntPtr)HT\_TOPLEFT; return; }

if (pos.X >= this.ClientSize.Width - BorderSize && pos.Y <= BorderSize) { m.Result = (IntPtr)HT\_TOPRIGHT; return; }

if (pos.X <= BorderSize && pos.Y >= this.ClientSize.Height - BorderSize) { m.Result = (IntPtr)HT\_BOTTOMLEFT; return; }

if (pos.X >= this.ClientSize.Width - BorderSize && pos.Y >= this.ClientSize.Height - BorderSize) { m.Result = (IntPtr)HT\_BOTTOMRIGHT; return; }

if (pos.X <= BorderSize) { m.Result = (IntPtr)HT\_LEFT; return; }

if (pos.X >= this.ClientSize.Width - BorderSize) { m.Result = (IntPtr)HT\_RIGHT; return; }

if (pos.Y <= BorderSize) { m.Result = (IntPtr)HT\_TOP; return; }

if (pos.Y >= this.ClientSize.Height - BorderSize) { m.Result = (IntPtr)HT\_BOTTOM; return; }

}

}

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

{

if (e.Button == MouseButtons.Left)

{

ReleaseCapture();

SendMessage(Handle, WM\_NCLBUTTONDOWN, HT\_CAPTION, 0);

}

}

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

{

wasHiddenManually = true;

this.WindowState = FormWindowState.Minimized;

}

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

{

Application.Exit();

}

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

{

this.Hide();

MainMenuWindow mainMenuWindow = new MainMenuWindow();

mainMenuWindow.Show();

mainMenuWindow.Activate();

}

private void toolStripMenuItem3\_Click(object sender, EventArgs e)

{

if (toolStripComboBox1.SelectedItem == "Множина Мандельброта")

{

RenderMandelbrot();

}

if (toolStripComboBox1.SelectedItem == "Множина Жуліа")

{

RenderJulia();

}

}

private void PictureBox1\_MouseWheel(object sender, MouseEventArgs e)

{

double zoomFactor = e.Delta > 0 ? 0.9 : 1.1;

double mouseRe = centerX + (e.X - pictureBox1.Width / 2.0) \* (scale / pictureBox1.Width);

double mouseIm = centerY + (e.Y - pictureBox1.Height / 2.0) \* (scale / pictureBox1.Height);

scale \*= zoomFactor;

centerX = mouseRe + (centerX - mouseRe) \* zoomFactor;

centerY = mouseIm + (centerY - mouseIm) \* zoomFactor;

if (toolStripComboBox1.SelectedItem == "Множина Мандельброта")

{

RenderMandelbrot();

}

if (toolStripComboBox1.SelectedItem == "Множина Жуліа")

{

RenderJulia();

}

}

private void RenderMandelbrot()

{

int width = pictureBox1.Width;

int height = pictureBox1.Height;

Bitmap bmp = new Bitmap(width, height);

for (int py = 0; py < height; py++)

{

for (int px = 0; px < width; px++)

{

double x0 = centerX + (px - width / 2.0) \* (scale / width);

double y0 = centerY + (py - height / 2.0) \* (scale / height);

double dx = x0 - 0.25;

double P = Math.Sqrt(dx \* dx + y0 \* y0);

double Om = Math.Atan2(y0, dx);

double Pc = 0.5 - 0.5 \* Math.Cos(Om);

Color color;

if (P <= Pc)

{

color = Color.Black;

}

else

{

double x = 0;

double y = 0;

int iteration = 0;

while (x \* x + y \* y <= 4 && iteration < maxIterations)

{

double xtemp = x \* x - y \* y + x0;

y = 2 \* x \* y + y0;

x = xtemp;

iteration++;

}

color = iteration == maxIterations

? Color.Black

: Color.FromArgb(255, iteration % 8 \* 32, iteration % 16 \* 16, iteration % 32 \* 8);

}

bmp.SetPixel(px, py, color);

}

}

pictureBox1.Image = bmp;

}

private void RenderJulia()

{

int width = pictureBox1.Width;

int height = pictureBox1.Height;

Bitmap bmp = new Bitmap(width, height);

double cRe = -0.7; //-0.7 // 0.355 // -0.8 // 0.285

double cIm = 0.27015; // 0.27015 // 0.355 // 0.156 // 0.01

Color color;

for (int py = 0; py < height; py++)

{

for (int px = 0; px < width; px++)

{

double x = centerX + (px - width / 2.0) \* (scale / width);

double y = centerY + (py - height / 2.0) \* (scale / height);

int iteration = 0;

while (x \* x + y \* y <= 4 && iteration < maxIterations)

{

double xtemp = x \* x - y \* y + cRe;

y = 2 \* x \* y + cIm;

x = xtemp;

iteration++;

}

color = iteration == maxIterations

? Color.Black

: Color.FromArgb(255, iteration % 8 \* 32, iteration % 16 \* 16, iteration % 32 \* 8);

bmp.SetPixel(px, py, color);

}

}

pictureBox1.Image = bmp;

}

private void toolStripTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

e.Handled = true;

}

private void toolStripTextBox1\_TextChanged(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(toolStripTextBox1.Text))

{

maxIterations = 0;

}

else

{

maxIterations = int.Parse(toolStripTextBox1.Text);

}

}

private void toHomeStripMenuItem\_Click\_1(object sender, EventArgs e)

{

this.Hide();

MainMenuWindow mainMenuWindow = new MainMenuWindow();

mainMenuWindow.Show();

mainMenuWindow.Activate();

}

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

{

if (pictureBox1.Image == null)

{

MessageBox.Show("Неможливо зберегти пусте зображення", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "PNG Image|\*.jpg",

Title = "Зберегти зображення",

};

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

{

using (Bitmap bm = new Bitmap(pictureBox1.Image))

{

bm.Save(saveFileDialog.FileName, ImageFormat.Jpeg);

}

}

}

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

if (!moreInformationTextBox.Visible)

{

moreInformationTextBox.Visible = true;

pictureBox1.Visible = false;

}

else

{

moreInformationTextBox.Visible = false;

pictureBox1.Visible = true;

}

}

1. Результат виконання програми

Малювання множини Мандеоьброта та Жуліа(див. Рисунок 31/32)

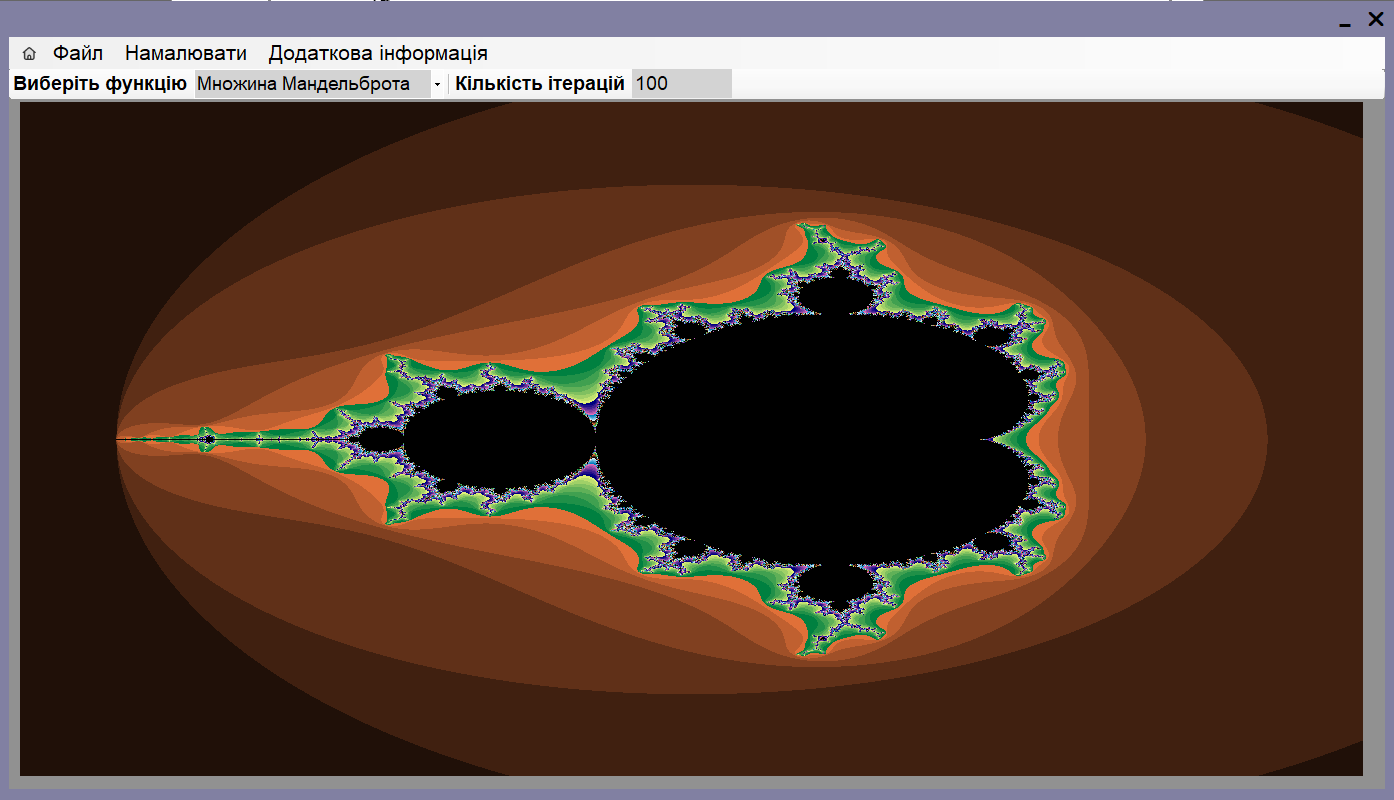


Рисунок 31 - Множина Мандельброта

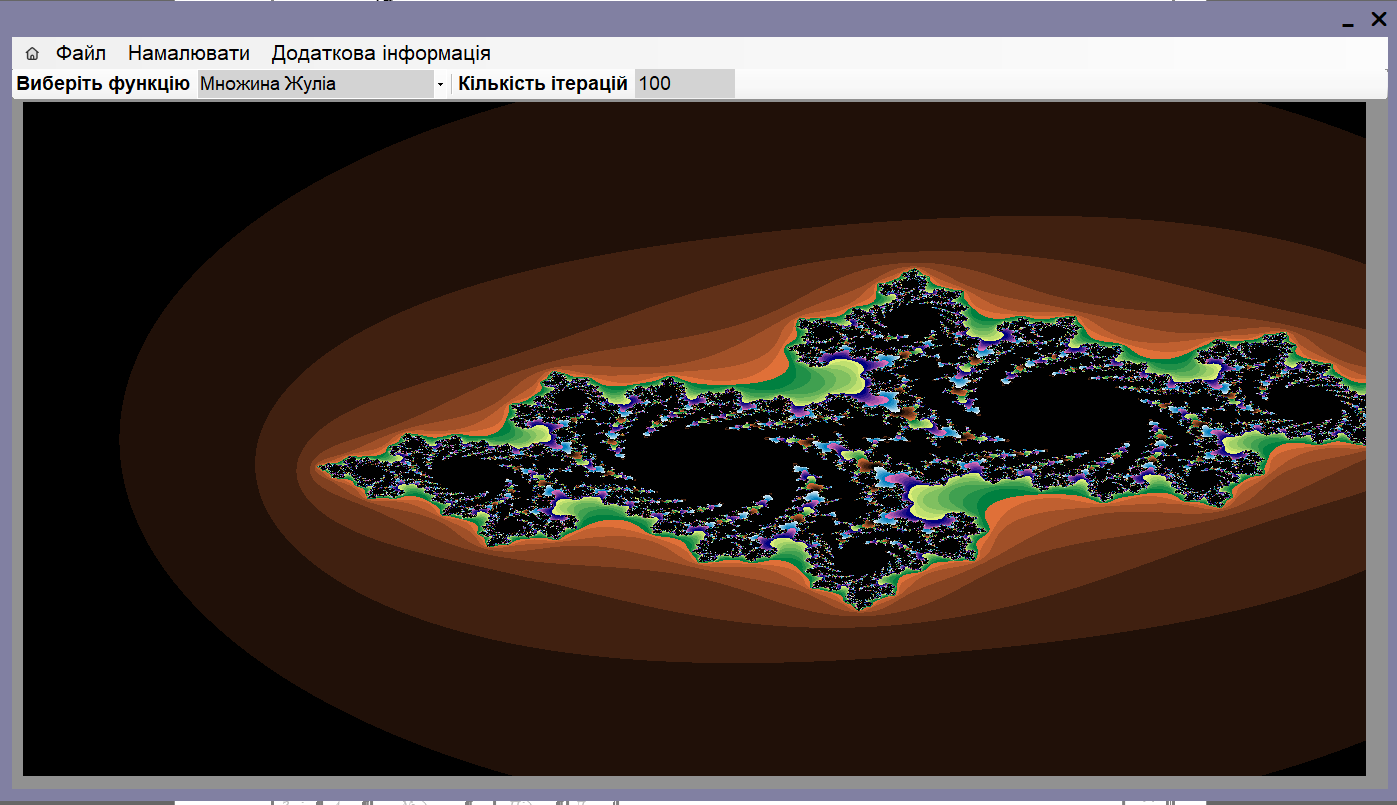


Рисунок 32 - Множина Жуліа

1. Виведення додаткової інформації(див. Рисунок 33)

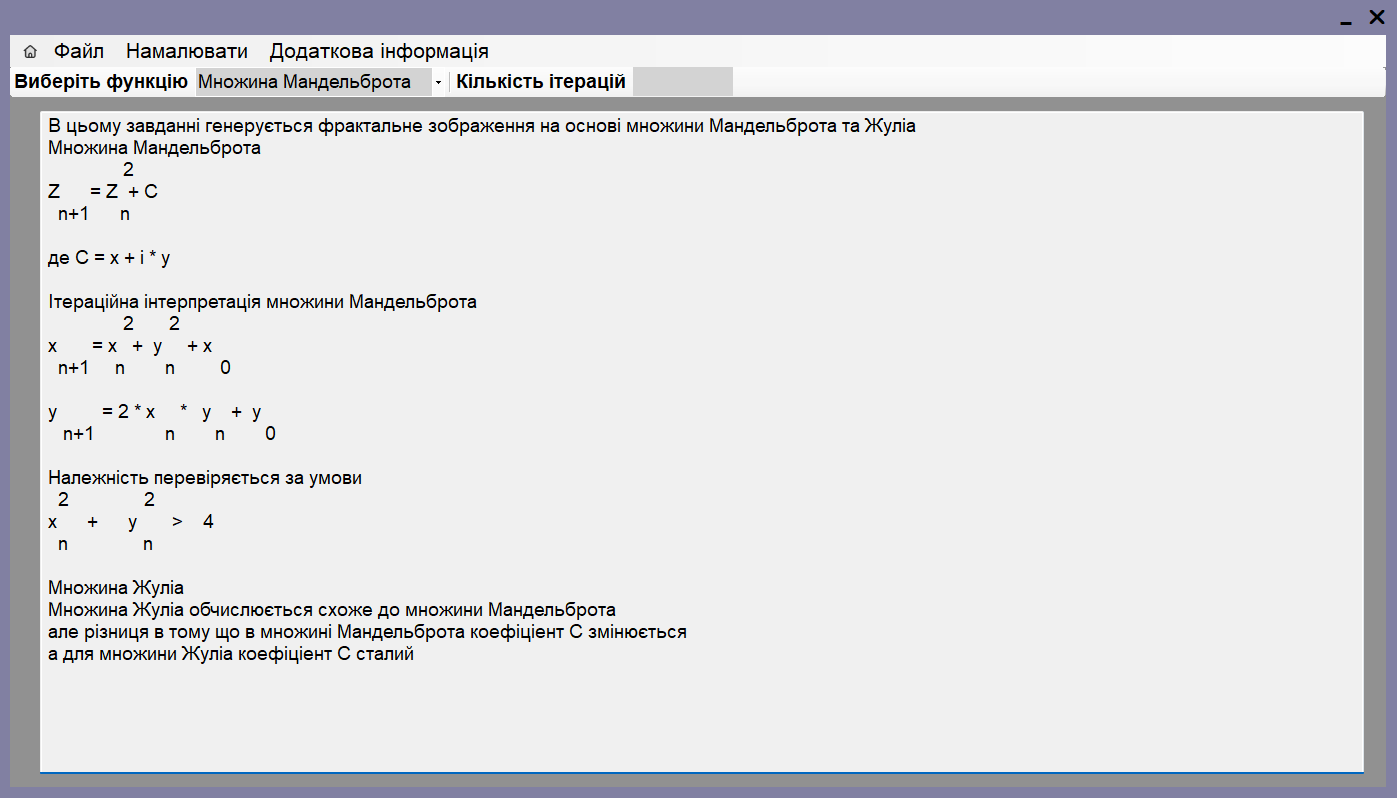


Рисунок 33 - Вікно додаткової інформації

Завдання 5 Робота зі структурою

1. Наступним кроком було створено нову форму та названо її TaskFiveWindow.
2. Після того як створили форму розміщуємо на ній об’єкти для завдання (див. Рисунок 34)

**Варіант 5 1. Описати структуру з ім'ям AEROFLOT, що містить наступні поля: • NAZN - назва пункту призначення рейсу; • NUMR - номер рейсу; • TIP - тип літака. 2. Написати програму, яка виконує наступні дії: • введення з клавіатури даних в бінарний файл AIRPORT, що складається з семи елементів типу AEROFLOT; записи повинні бути розміщені в алфавітному порядку за назвами пунктів призначення; читання записів файлу на екран у вигляді таблиці; • виведення на екран пунктів призначення і номерів рейсів, що обслуговуються літаком, тип якого введено з клавіатури; • якщо таких рейсів немає, вивести відповідне повідомлення.**



Рисунок 34 – Проектування форми

1. Відформатуємо форму для результату(див. Рисунок 35)

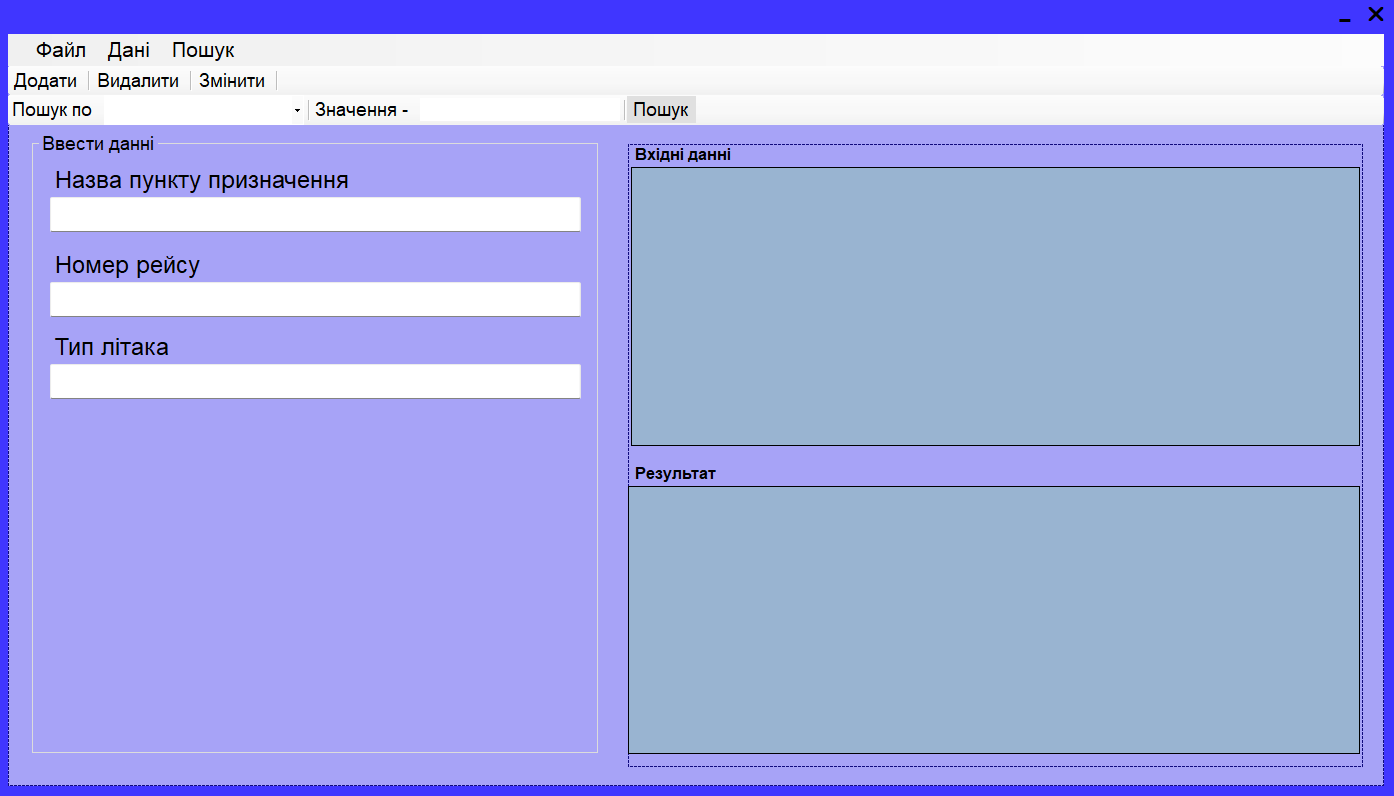


Рисунок 35 – Кінцевий варіант форми

1. Написано відповідний програмний код до форми в просторі імен ІндивідуальнеЗвданняCSharp.
2. Створено змінні та конструктор для форми.

#region Змінні для дизайну форми

private const int WM\_NCLBUTTONDOWN = 0xA1;

private const int HT\_CAPTION = 0x2;

[DllImport("user32.dll")]

public static extern int SendMessage(IntPtr hWnd, int Msg, int wParam, int lParam);

[DllImport("user32.dll")]

public static extern bool ReleaseCapture();

private const int HT\_LEFT = 10;

private const int HT\_RIGHT = 11;

private const int HT\_TOP = 12;

private const int HT\_TOPLEFT = 13;

private const int HT\_TOPRIGHT = 14;

private const int HT\_BOTTOM = 15;

private const int HT\_BOTTOMLEFT = 16;

private const int HT\_BOTTOMRIGHT = 17;

private const int BorderSize = 8; // Ширина зони, за яку можна "тягнути" форму

private bool wasHiddenManually = false;

#endregion

#region Змінні для завдання

struct AEROFLOT

{

string NAZN;

string NUMR;

string TIP;

}

private List<AEROFLOT> aeroflotList = new List<AEROFLOT>();

private DataTable baseDataTable = new DataTable("AEROFLOTdata");

#endregion

public TaskFiveWindow()

{

InitializeComponent();

string relativePath = Path.Combine("Icons", "home-alt.svg");

string fullPath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, relativePath);

if (!File.Exists(fullPath))

{

MessageBox.Show($"Файл не знайдено:\n{fullPath}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

SvgDocument svgDoc = SvgDocument.Open(fullPath);

toHomeStripMenuItem.Image = svgDoc.Draw(26, 26);

this.DoubleBuffered = true;

this.SetStyle(ControlStyles.ResizeRedraw, true);

searchParamToolStripComboBox.SelectedItem = "Номеру рейсу";

inputLabelIOPanem.Location = new System.Drawing.Point(3, 1);

inputDataGridView.Location = new System.Drawing.Point(3, inputLabelIOPanem.Size.Height);

resLabelIOPanem.Location = new System.Drawing.Point(3, inputOutputPanel.Size.Height / 2);

resultDataGridView.Location = new System.Drawing.Point(3, resLabelIOPanem.Size.Height + resLabelIOPanem.Location.Y);

baseDataTable.Columns.Add("Місце прибуття", typeof(string));

baseDataTable.Columns.Add("Номер рейсу", typeof(string));

baseDataTable.Columns.Add("Тип літака", typeof(string));

inputDataGridView.DataSource = baseDataTable;

}

1. Перевизначено системну функцію WinProc для власного масштабування та перетягування форми

protected override void WndProc(ref Message m)

{

const int WM\_SYSCOMMAND = 0x0112;

const int SC\_RESTORE = 0xF120;

if (m.Msg == WM\_SYSCOMMAND && m.WParam.ToInt32() == SC\_RESTORE)

{

if (wasHiddenManually)

{

wasHiddenManually = false;

this.WindowState = FormWindowState.Normal;

Rectangle screen = Screen.FromControl(this).WorkingArea;

int x = (screen.Width - this.Width) / 2 + screen.Left;

int y = (screen.Height - this.Height) / 2 + screen.Top;

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

this.Opacity = 1;

this.Enabled = true;

}

base.WndProc(ref m);

return;

}

base.WndProc(ref m);

if (m.Msg == 0x84) // WM\_NCHITTEST

{

Point pos = new Point(m.LParam.ToInt32());

pos = this.PointToClient(pos);

if (pos.X <= BorderSize && pos.Y <= BorderSize) { m.Result = (IntPtr)HT\_TOPLEFT; return; }

if (pos.X >= this.ClientSize.Width - BorderSize && pos.Y <= BorderSize) { m.Result = (IntPtr)HT\_TOPRIGHT; return; }

if (pos.X <= BorderSize && pos.Y >= this.ClientSize.Height - BorderSize) { m.Result = (IntPtr)HT\_BOTTOMLEFT; return; }

if (pos.X >= this.ClientSize.Width - BorderSize && pos.Y >= this.ClientSize.Height - BorderSize) { m.Result = (IntPtr)HT\_BOTTOMRIGHT; return; }

if (pos.X <= BorderSize) { m.Result = (IntPtr)HT\_LEFT; return; }

if (pos.X >= this.ClientSize.Width - BorderSize) { m.Result = (IntPtr)HT\_RIGHT; return; }

if (pos.Y <= BorderSize) { m.Result = (IntPtr)HT\_TOP; return; }

if (pos.Y >= this.ClientSize.Height - BorderSize) { m.Result = (IntPtr)HT\_BOTTOM; return; }

}

}

1. Оброблено події для переміщення, зміни розміру, закриття та сховання форми

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

{

if (e.Button == MouseButtons.Left)

{

ReleaseCapture();

SendMessage(Handle, WM\_NCLBUTTONDOWN, HT\_CAPTION, 0);

}

}

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

{

wasHiddenManually = true;

this.WindowState = FormWindowState.Minimized;

}

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

{

Application.Exit();

}

private void TaskFiveWindow\_Resize(object sender, EventArgs e)

{

//------------ Для панельки --------------

inputLabelIOPanem.Location = new System.Drawing.Point(3, 1);

inputDataGridView.Location = new System.Drawing.Point(3, inputLabelIOPanem.Size.Height);

resLabelIOPanem.Location = new System.Drawing.Point(3, inputOutputPanel.Size.Height / 2);

resultDataGridView.Location = new System.Drawing.Point(3, resLabelIOPanem.Size.Height + resLabelIOPanem.Location.Y);

inputDataGridView.Size = new Size(inputOutputPanel.Size.Width - 3, (int)(inputOutputPanel.Size.Height \* 0.45));

resultDataGridView.Size = new Size(inputOutputPanel.Size.Width - 3, (int)(inputOutputPanel.Size.Height \* 0.45));

//------------ Для панельки --------------

//------------ Для вхідних данних --------------

label1.Location = new System.Drawing.Point(18, 35);

nameOfDestinationTextBox.Location = new System.Drawing.Point(label1.Location.X, label1.Location.Y + label1.Size.Height + 6);

label2.Location = new System.Drawing.Point(nameOfDestinationTextBox.Location.X, nameOfDestinationTextBox.Location.Y + nameOfDestinationTextBox.Size.Height + 20);

flightNumberTextBox.Location = new System.Drawing.Point(label2.Location.X, label2.Location.Y + label2.Size.Height + 6);

label3.Location = new System.Drawing.Point(flightNumberTextBox.Location.X, flightNumberTextBox.Location.Y + flightNumberTextBox.Size.Height + 20);

planeTypeTextBox.Location = new System.Drawing.Point(label3.Location.X, label3.Location.Y + label3.Size.Height + 6);

//------------ Для вхідних данних --------------

}

1. Оброблено події для перевірки вхідних данних

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

{

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

e.Handled = true;

}

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

{

if (!char.IsDigit(e.KeyChar) && !char.IsUpper(e.KeyChar) && !char.IsControl(e.KeyChar))

e.Handled = true;

}

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

{

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

e.Handled = true;

if (e.KeyChar == '-' && (sender as TextBox).Text.Contains('-'))

e.Handled = true;

}

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

{

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

e.Handled = true;

if (e.KeyChar == '-' && (sender as TextBox).Text.Contains('-'))

e.Handled = true;

}

1. Оброблено події натискання кнопок в головному меню

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

{

this.Hide();

MainMenuWindow mainMenuWindow = new MainMenuWindow();

mainMenuWindow.Show();

mainMenuWindow.Activate();

}

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

{

dataToolStrip.Visible = true;

toolStrip1.Visible = false;

}

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

{

dataToolStrip.Visible = false;

toolStrip1.Visible = true;

}

1. Написано функцію для пошуку інформації по полю

private DataTable searchInformation(string fieldName, string value)

{

DataTable resultTable = baseDataTable.Clone();

foreach (DataRow row in baseDataTable.Rows)

{

if (!baseDataTable.Columns.Contains(fieldName))

{

MessageBox.Show($"Колонка '{fieldName}' не існує.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return resultTable;

}

if (row[fieldName].ToString().Equals(value, StringComparison.OrdinalIgnoreCase))

{

resultTable.ImportRow(row);

}

}

return resultTable;

}

1. Оброблено подію при натисканні кнопки Пошук

private void toolStripButton1\_Click(object sender, EventArgs e)

{

if (string.IsNullOrWhiteSpace(valueSortToolStripTextBox.Text))

{

MessageBox.Show("Будь ласка, введіть значення для пошуку.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string searchField = "";

if (searchParamToolStripComboBox.SelectedItem == "Пункту призначення")

searchField = "Місце прибуття";

else if (searchParamToolStripComboBox.SelectedItem == "Номеру рейсу")

searchField = "Номер рейсу";

else if (searchParamToolStripComboBox.SelectedItem == "Типу літака")

searchField = "Тип літака";

else

{

MessageBox.Show("Будь ласка, виберіть правильний параметр пошуку.");

return;

}

DataTable result = searchInformation(searchField, valueSortToolStripTextBox.Text);

if (result.Rows.Count == 0)

{

MessageBox.Show("Записи за вказаним критерієм не знайдено.", "Результат пошуку", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

resultDataGridView.DataSource = result;

}

1. Оброблено події при натисканні кнопки Відкрити та Зберегти в вкладці Файл

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

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Database file|\*.data",

Title = "Відкрити файл .data"

};

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

{

try

{

string filePath = openFileDialog.FileName;

baseDataTable.Clear();

using (StreamReader reader = new StreamReader(filePath))

{

string line;

while ((line = reader.ReadLine()) != null)

{

string[] parts = line.Split(';');

if (parts.Length != 3)

continue;

baseDataTable.Rows.Add(parts[0].Trim(), parts[1].Trim(), parts[2].Trim());

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при відкритті файлу: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "Database file|\*.data",

Title = "Зберегти файл .data"

};

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

{

try

{

string filePath = saveFileDialog.FileName;

using (StreamWriter writer = new StreamWriter(filePath, false, Encoding.UTF8))

{

foreach (DataRow row in baseDataTable.Rows)

{

if (row.IsNull("Місце прибуття") || row.IsNull("Номер рейсу") || row.IsNull("Тип літака"))

continue;

string destination = row["Місце прибуття"].ToString().Trim();

string flightNumber = row["Номер рейсу"].ToString().Trim();

string planeType = row["Тип літака"].ToString().Trim();

if (string.IsNullOrWhiteSpace(destination) || string.IsNullOrWhiteSpace(flightNumber) || string.IsNullOrWhiteSpace(planeType))

continue;

string line = $"{destination};{flightNumber};{planeType}";

writer.WriteLine(line);

}

}

MessageBox.Show("Дані успішно збережено.", "Успіх", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Помилка при збереженні файлу: {ex.Message}", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

1. Протестовано роботу програми

Додавання рейсу(див. Рисунок 36/37)

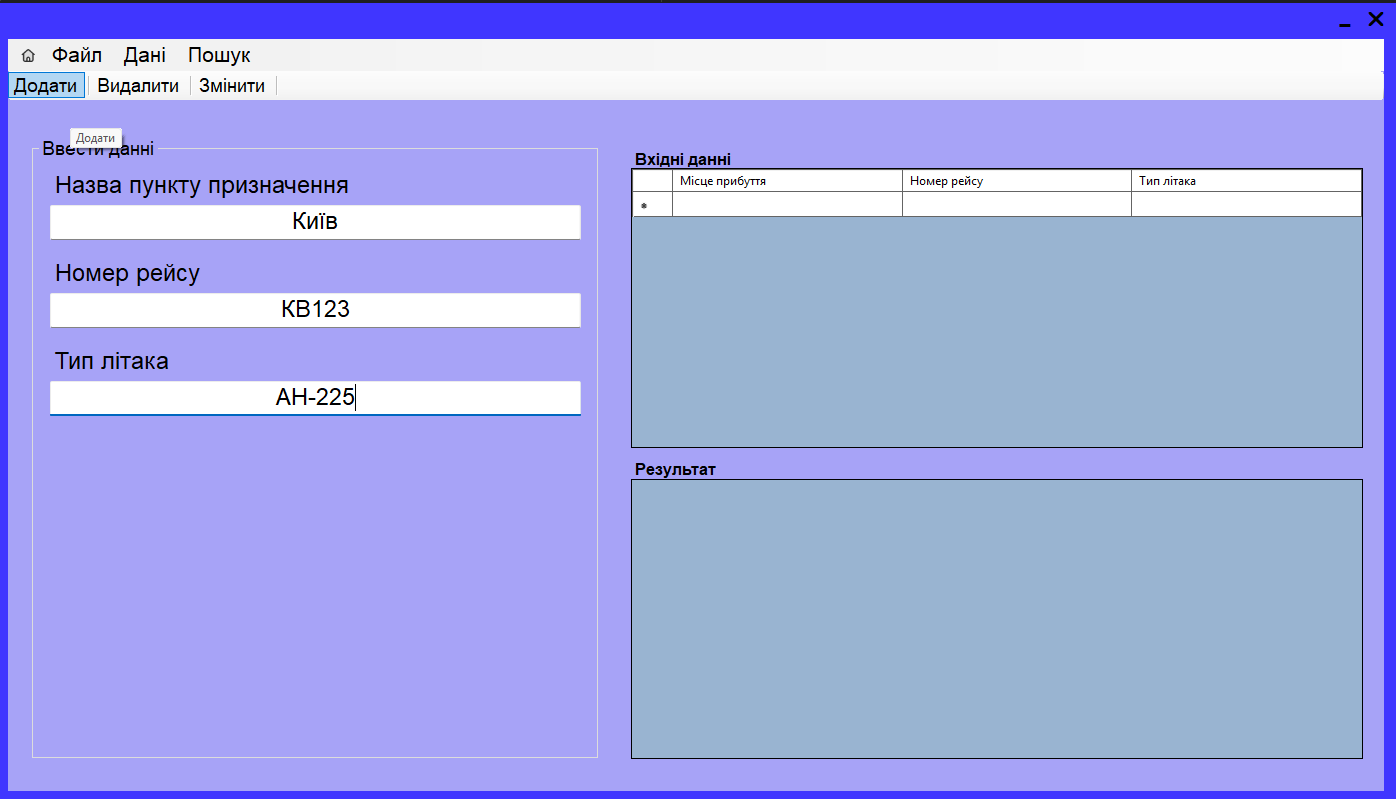


Рисунок 36 – Додавання данних в поле

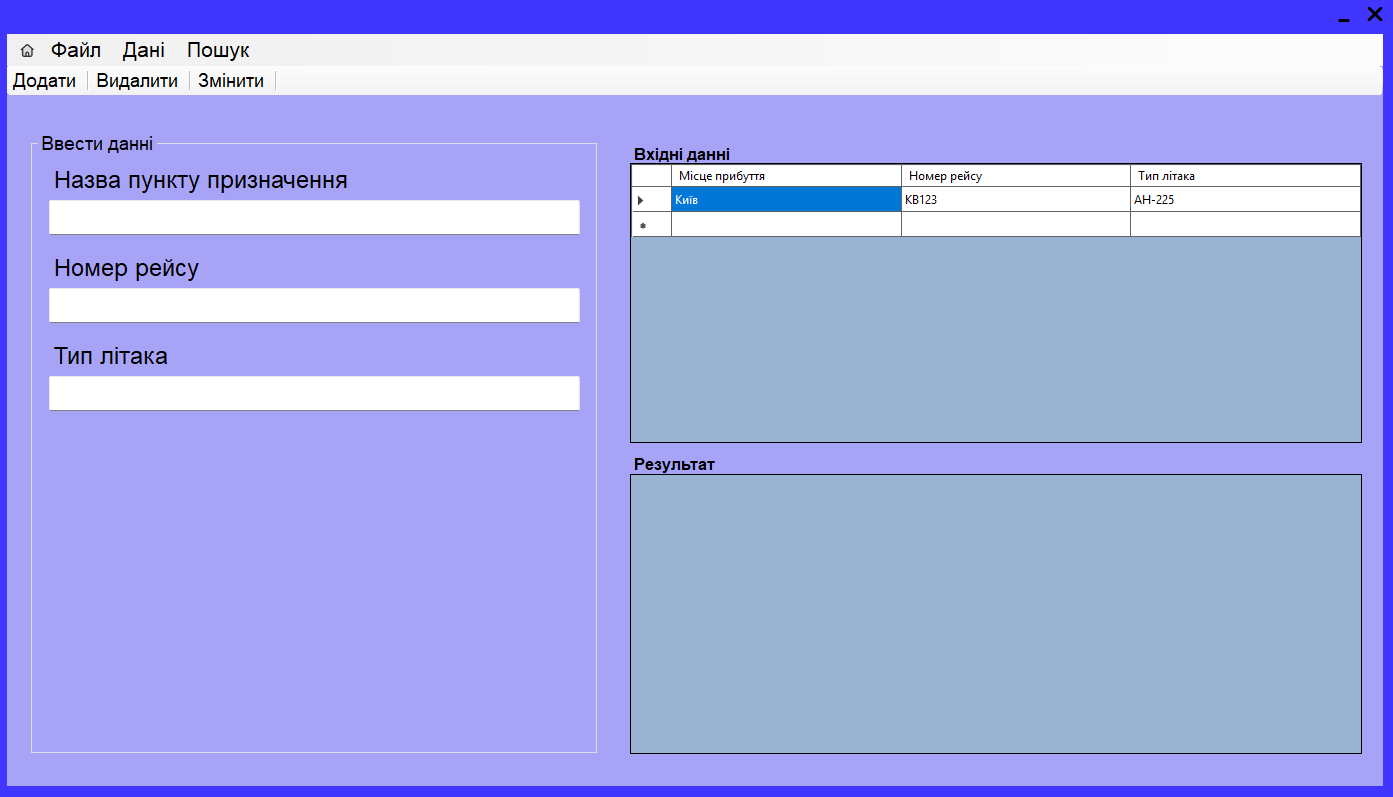


Рисунок 37 – Данні в DataGreedView

1. Видалення данних(див. Рисунок 38/39/40)

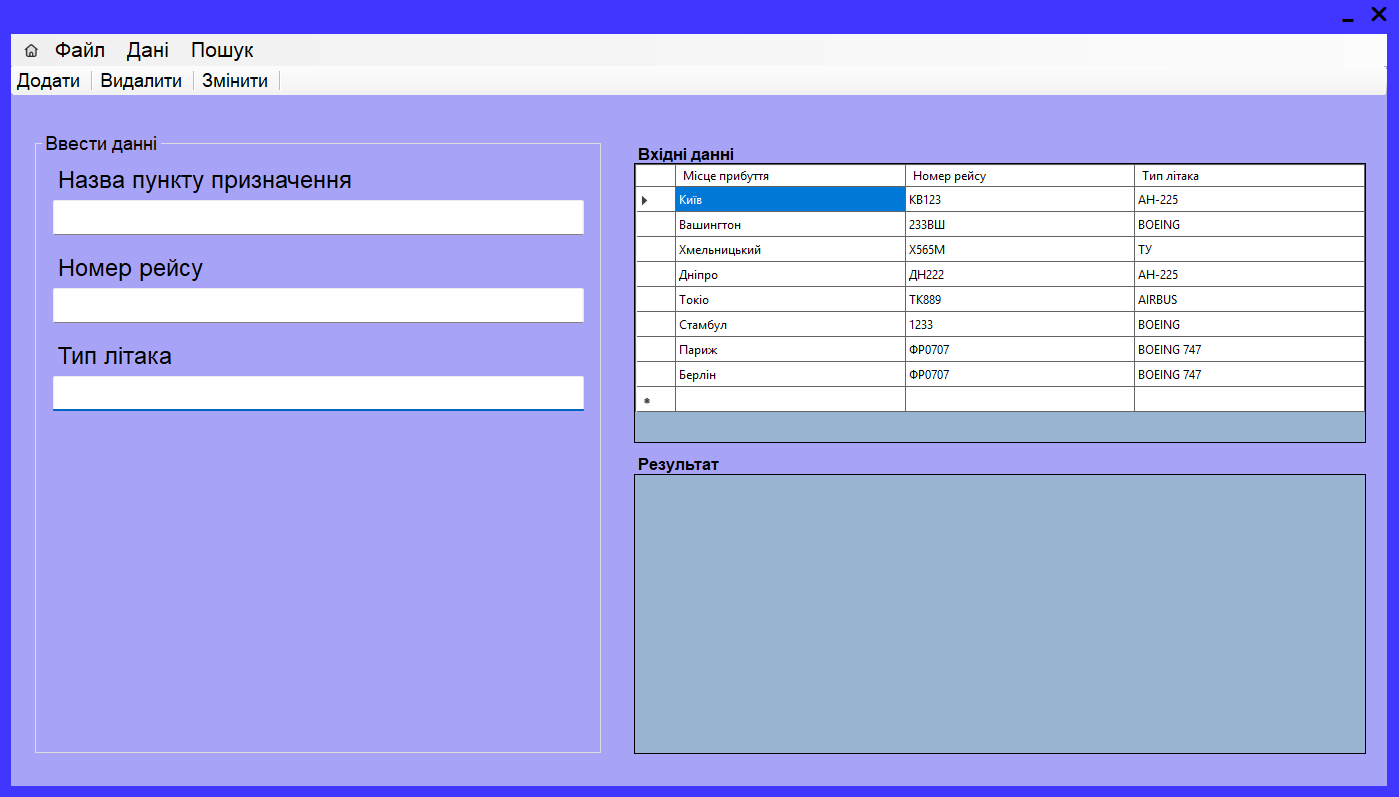


Рисунок 38 – Данні до видалення

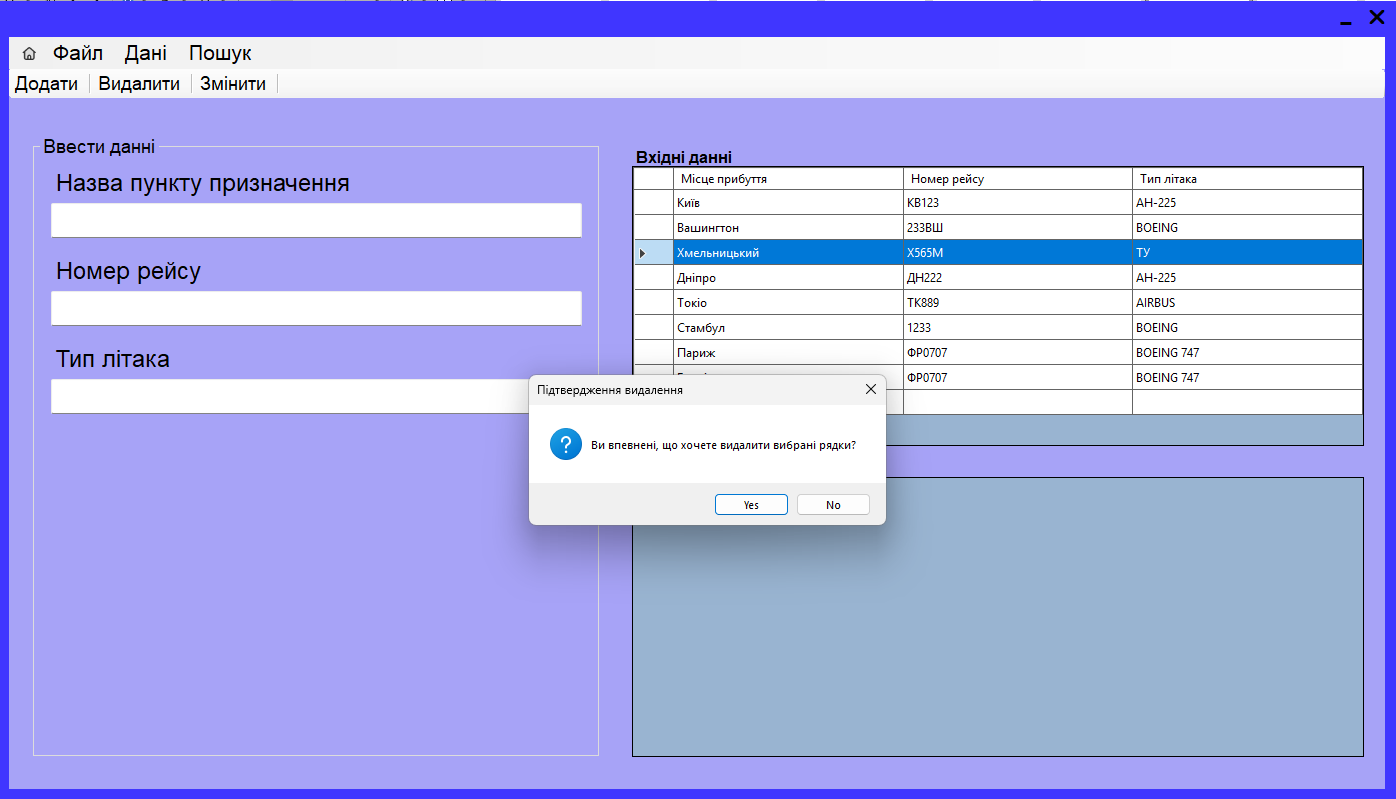


Рисунок 39 – Перевірка чи точно користувач хоче видалити данні

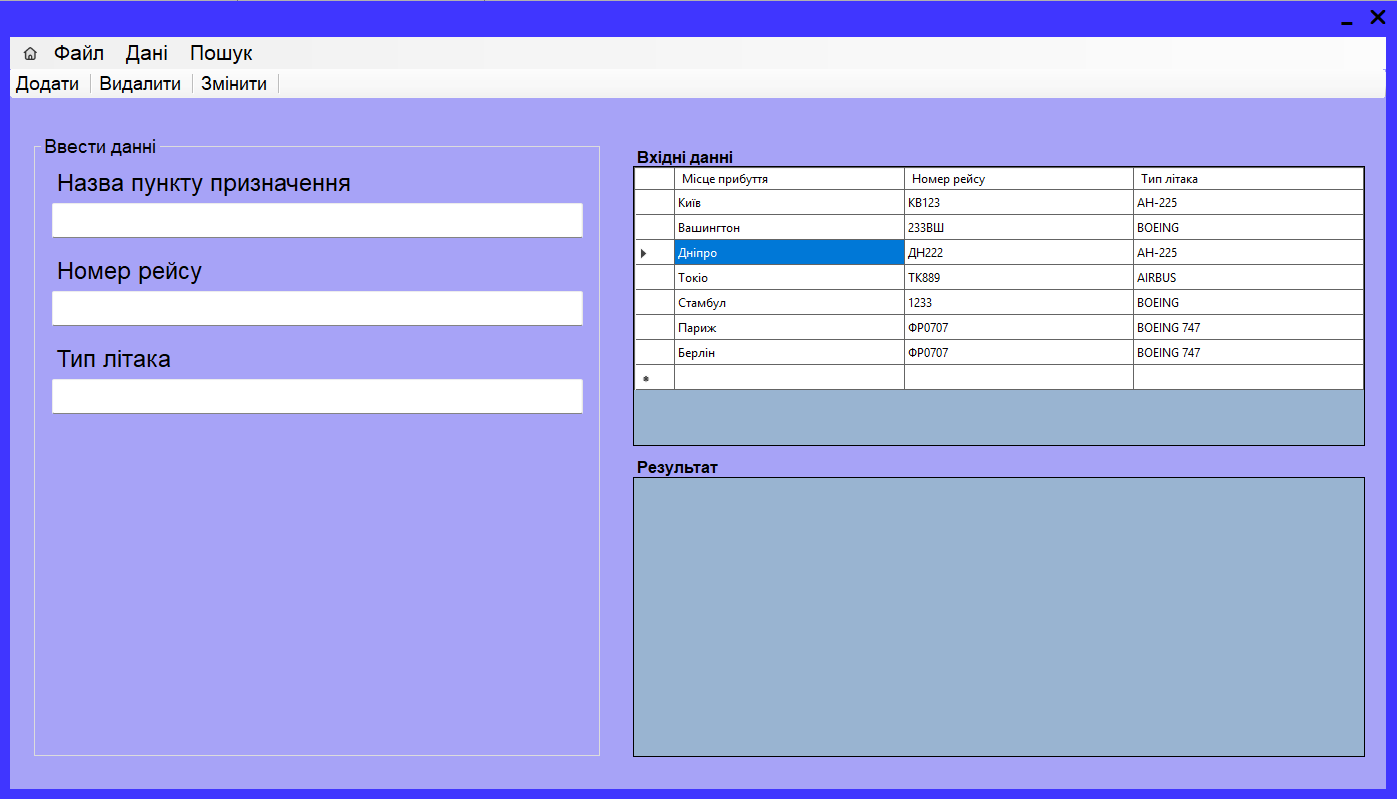


Рисунок 40 – Данні після видалення

1. Пошук данних по заданому полю(див. Рисунок 41)

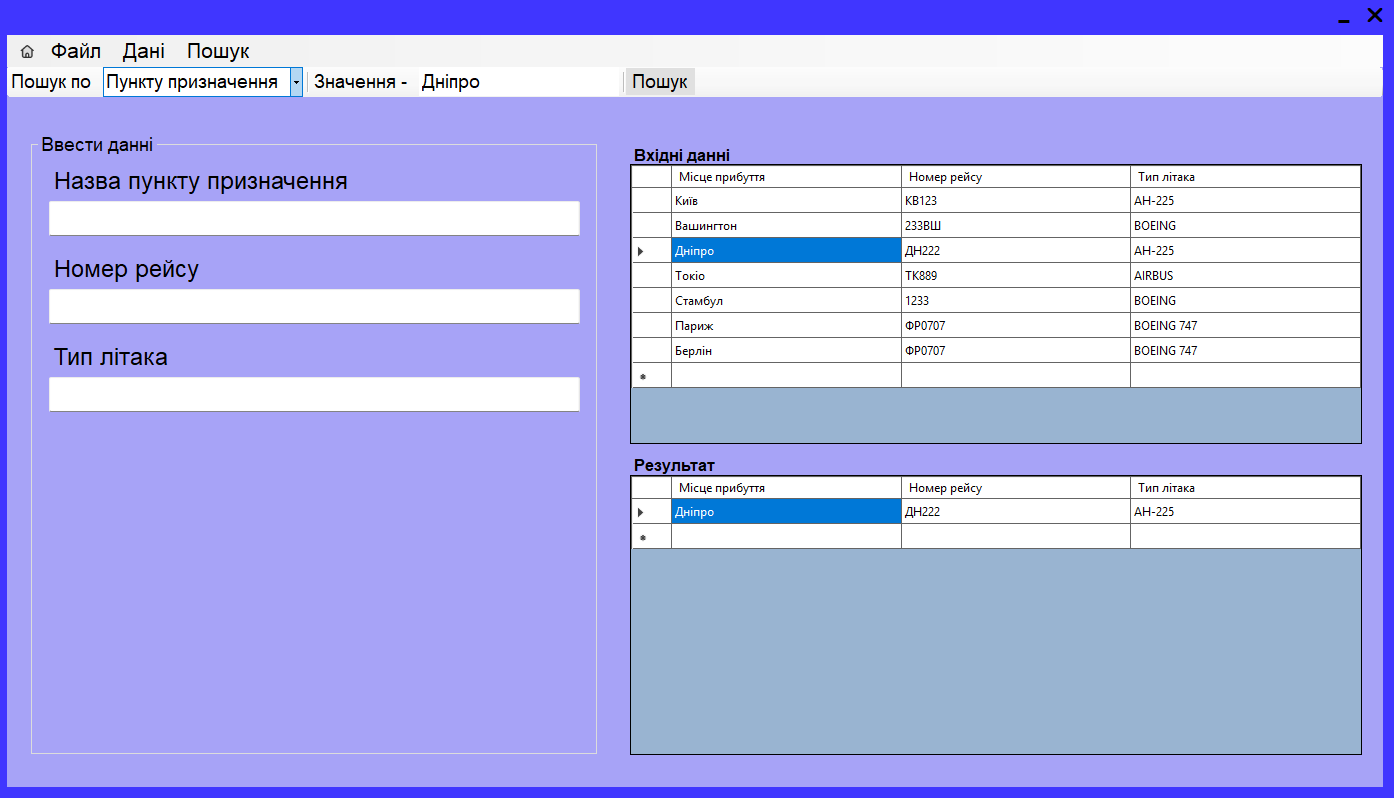


Рисунок 41 – Пошук по полю Пункт призначення

1. Перевірка коли немає збігів (див. Рисунок 42)

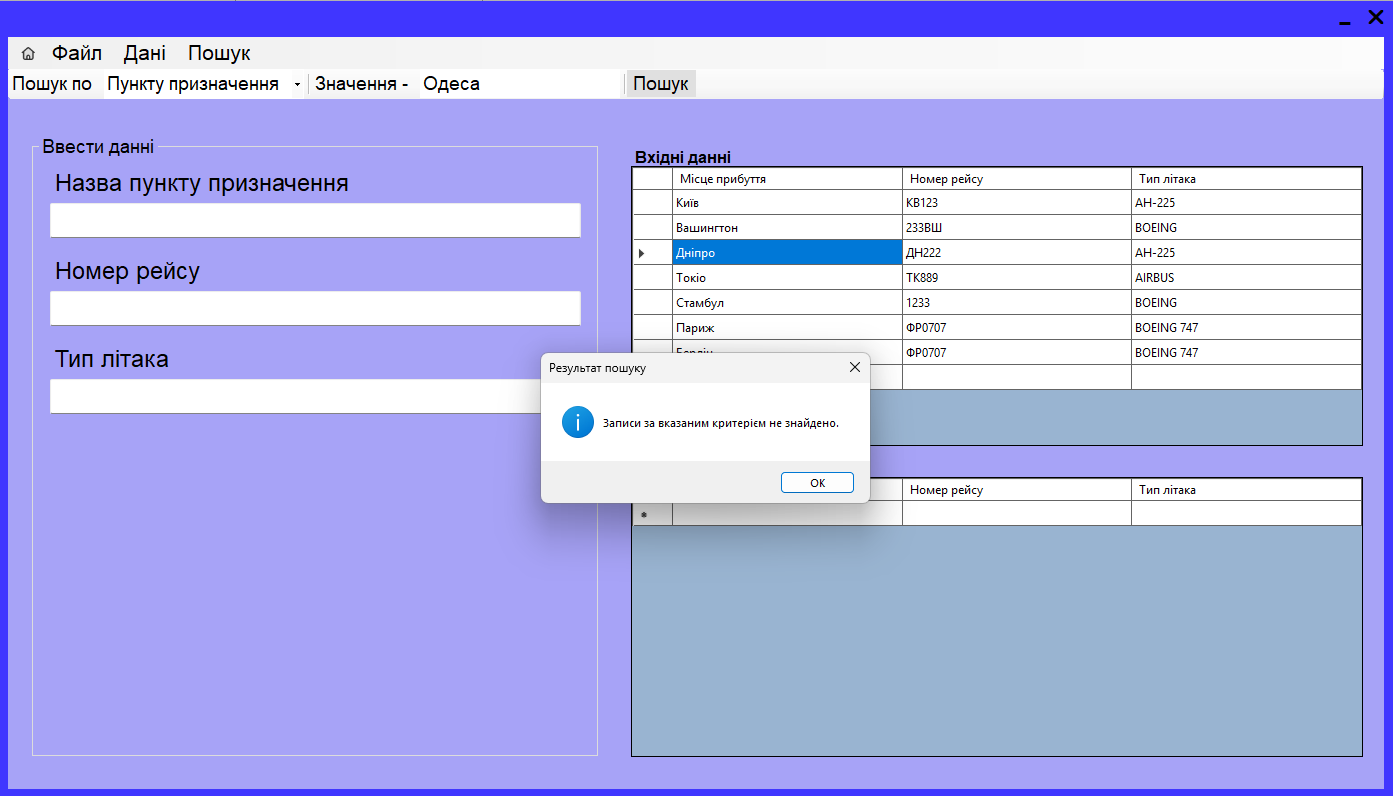


Рисунок 42 – Немає збігів

1. Збереження в файл(див. Рисунок 43)

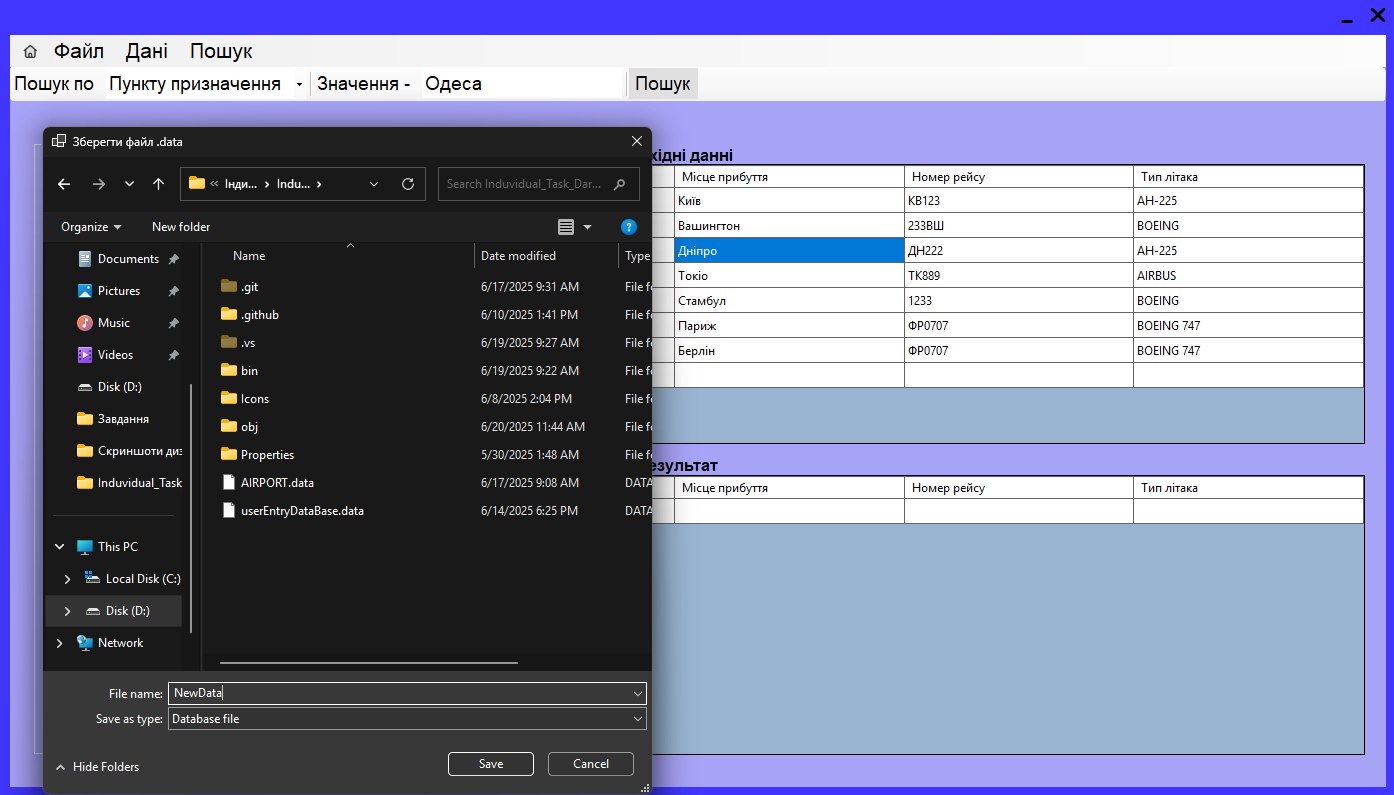


Рисунок 43 – Збереження в файл

1. Відкриття файлу(див. Рисунок 44)

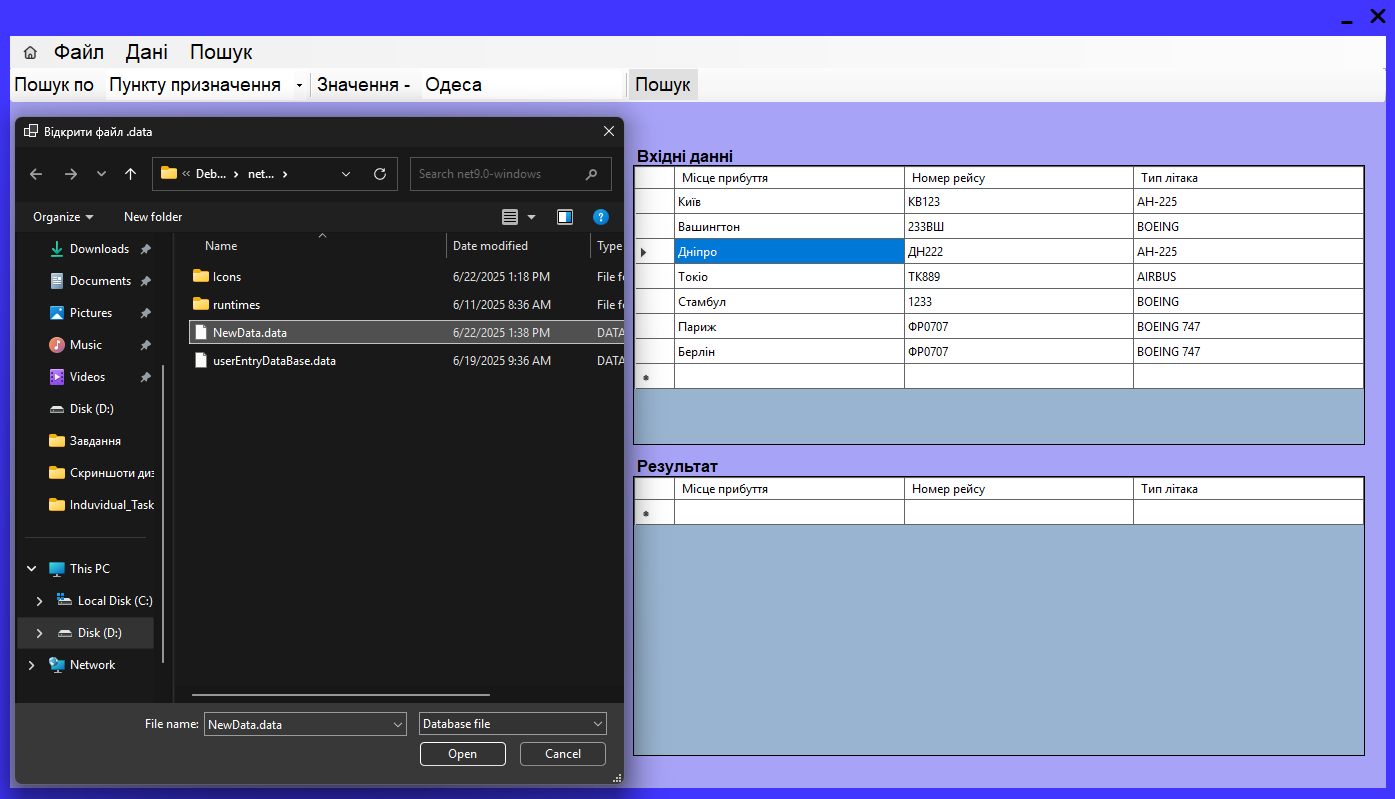


Рисунок 44 – Відкриття файлу

Висновок

У результаті виконання роботи було засвоєно базові принципи розробки програм з графічним інтерфейсом у середовищі Visual Studio з використанням технології Windows Forms. Отримано практичні навички створення форм, додавання елементів керування (кнопок, текстових полів, таблиць тощо), обробки подій користувача та реалізації взаємодії між компонентами. Також розглянуто основи структури проєкту WinForms і способи організації логіки програми.