МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Криворізький національний університет
Кафедра моделювання та програмного забезпечення

Звіт
з лабораторної роботи №3

УПРАВЛІННЯ ТАБЛИЧНИМИ ДАНИМИ У ВІЗУАЛЬНИХ ДОДАТКАХ

Студент групи ІПЗ-21-2
Губарєв Р.В.
+380980190289
Викладачі
Козиков А. В.
Гриценко А. М.

Кривий Ріг
2022

1. **Короткі теоретичні відомості про візуальні об’єкти (компонентів), що використовуються.**

ComboBox - Відображає список елементів, що розкривається.

Label - Відображає недоступний для безпосередньої зміни користувачем текст.

MenuStrip - Створює меню, що настроюється.

DataGridView - Елемент керування DataGridView надає таблицю, що настроюється, для відображення даних. Клас DataGridView дозволяє налаштовувати комірки, рядки, стовпці та межі.

Button - Запускає, зупиняє чи перериває процес.

2. **Короткі теоретичні відомості про управління даними у таблицях, навести приклади використання**

DataGridView можна використовувати для відображення даних з базовим джерелом даних або без нього. Без вказівки джерела даних можна створювати стовпці та рядки, що містять дані, і додавати їх безпосередньо до DataGridView за допомогою Rows і Columns. Також можна використовувати колекцію Rows для доступу до DataGridViewRow об'єктів та DataGridViewRow.Cells властивості для читання або запису значень осередків безпосередньо. Індексатор Item[] також надає прямий доступ до осередків.

3. **Вихідний текст програми**

Form1.cs

```
using System.Windows.Forms;
using System.IO;
using System.Data;
using System.ComponentModel;
using System.Collections;

namespace Lab3
{
    public partial class Form1 : Form
    {
        private Dictionary<DataGridViewCell, object> hiddenData = new
Dictionary<DataGridViewCell, object>();
        private DataGridView activeDataGridView; // Змінна для збереження
поточної активної таблиці

        public Form1()
        {
            InitializeComponent();
            activeDataGridView = dataGridView1; // Початково активна таблиця -
dataGridView1

        }
```

```
        private void редагуватиToolStripMenuItem_Click(object sender,
EventArgs e)
        {

        }

        private void завантажитиToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "Text Files|*.txt";
            openFileDialog.Title = "Виберіть текстовий файл";

            if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                string filePath = openFileDialog.FileName;

                if (dataGridView1.Focused)
                {
                    LoadDataToDataGridView(filePath, dataGridView1);
                }
                else if (dataGridView2.Focused)
                {
                    LoadDataToDataGridView(filePath, dataGridView2);
                }
            }
        }
        private void LoadDataToDataGridView(string filePath, DataGridView
dataGridView)
        {
            string[] lines = File.ReadAllLines(filePath);

            dataGridView.Rows.Clear();

            for (int i = 0; i < lines.Length; i++)
            {
                string line = lines[i];
                string[] values = line.Split(' ');

                dataGridView.Rows.Add(values);
            }
        }

        private void зберегтиToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            if (!IsDataGridViewEmpty(dataGridView1))
            {
                SaveDataToFile(dataGridView1, "H:\\University\\2 курс\\2
семестр\\ООП\\Код\\Lab3\\file1.txt");
            }

            if (!IsDataGridViewEmpty(dataGridView2))
            {
                SaveDataToFile(dataGridView2, "H:\\University\\2 курс\\2
семестр\\ООП\\Код\\Lab3\\file2.txt");
            }

            if (!IsDataGridViewEmpty(dataGridView3))
            {
                SaveDataToFile(dataGridView3, "H:\\University\\2 курс\\2
семестр\\ООП\\Код\\Lab3\\file3.txt");
            }
        }
```

```
        private bool IsDataGridViewEmpty(DataGridView dataGridView)
        {
            return dataGridView.Rows.Count == 0;
        }

        private void SaveDataToFile(DataGridView dataGridView, string
filePath)
        {
            using (StreamWriter writer = new StreamWriter(filePath))
            {
                for (int i = 0; i < dataGridView.Rows.Count; i++)
                {
                    for (int j = 0; j < dataGridView.Columns.Count; j++)
                    {
                        DataGridViewCell cell = dataGridView.Rows[i].Cells[j];
                        string value = cell.Value?.ToString() ?? string.Empty;
                        writer.Write(value);

                        if (j < dataGridView.Columns.Count - 1)
                        {
                            writer.Write(" ");
                        }
                    }

                    writer.WriteLine();
                }
            }
        }

        private void змінитиКолірВмістуТаблиць ToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            ColorDialog colorDialog = new ColorDialog();
            if (colorDialog.ShowDialog() == DialogResult.OK)
            {
                Color selectedColor = colorDialog.Color;
                ApplyTextColor(selectedColor);
            }
        }
        private void ApplyTextColor(Color color)
        {
            if (dataGridView1.Focused)
            {
                if (dataGridView1.SelectedCells.Count > 0)
                {
                    foreach (DataGridViewCell cell in
dataGridView1.SelectedCells)
                    {
                        cell.Style.ForeColor = color;
                    }
                }
            }
            else if (dataGridView2.Focused)
            {
                if (dataGridView2.SelectedCells.Count > 0)
                {
                    foreach (DataGridViewCell cell in
dataGridView2.SelectedCells)
                    {
                        cell.Style.ForeColor = color;
                    }
                }
            }

        }
```

```
        private void змінитиКолірТаблицьToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            ColorDialog colorDialog = new ColorDialog();
            if (colorDialog.ShowDialog() == DialogResult.OK)
            {
                Color selectedColor = colorDialog.Color;
                ApplyCellBackgroundColor(selectedColor);
            }
        }
        private void ApplyCellBackgroundColor(Color color)
        {
            if (dataGridView1.Focused)
            {
                if (dataGridView1.SelectedCells.Count > 0)
                {
                    foreach (DataGridViewCell cell in
dataGridView1.SelectedCells)
                    {
                        cell.Style.BackColor = color;
                    }
                }
            }
            else if (dataGridView2.Focused)
            {
                if (dataGridView2.SelectedCells.Count > 0)
                {
                    foreach (DataGridViewCell cell in
dataGridView2.SelectedCells)
                    {
                        cell.Style.BackColor = color;
                    }
                }
            }
        }

        private void приховатиДаніToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            HideData();
        }
        private void HideData()
        {
            if (dataGridView1.Focused)
            {
                foreach (DataGridViewCell cell in dataGridView1.SelectedCells)
                {
                    if (!hiddenData.ContainsKey(cell))
                    {
                        hiddenData.Add(cell, cell.Value);
                        cell.Value = string.Empty;
                        cell.Style.ForeColor = Color.Gray;
                    }
                }
            }
            else if (dataGridView2.Focused)
            {
                foreach (DataGridViewCell cell in dataGridView2.SelectedCells)
                {
                    if (!hiddenData.ContainsKey(cell))
                    {
                        hiddenData.Add(cell, cell.Value);
                        cell.Value = string.Empty;
                        cell.Style.ForeColor = Color.Gray;
```

```
                    }
                }
            }
        }

        private void показатиДаніToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            ShowHiddenCellsData();
        }
        private void ShowHiddenCellsData()
        {
            if (dataGridView1.Focused)
            {
                foreach (KeyValuePair<DataGridViewCell, object> pair in
hiddenData)
                {
                    DataGridViewCell cell = pair.Key;
                    cell.Value = pair.Value;
                    cell.Style.ForeColor =
dataGridView1.DefaultCellStyle.ForeColor;
                }
                hiddenData.Clear();
            }
            if (dataGridView2.Focused)
            {
                foreach (KeyValuePair<DataGridViewCell, object> pair in
hiddenData)
                {
                    DataGridViewCell cell = pair.Key;
                    cell.Value = pair.Value;
                    cell.Style.ForeColor =
dataGridView2.DefaultCellStyle.ForeColor;
                }
                hiddenData.Clear();
            }
        }

        private void сортуванняЗаСтовпцямиToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            SortSelectedColumn();
        }
        private void SortSelectedColumn()
        {
            if (dataGridView1.Focused)
            {
                if (dataGridView1.SortOrder == SortOrder.Ascending)
                {
                    // Якщо DataGridView вже сортується за вибраною колонкою в
порядку зростання,
                    // змінюємо порядок сортування на спадання

dataGridView1.Sort(dataGridView1.Columns[dataGridView1.CurrentCell.ColumnIndex
], System.ComponentModel.ListSortDirection.Descending);
                }
                else
                {
                    // Якщо DataGridView сортується за вибраною колонкою в
порядку спадання або не сортується взагалі,
                    // змінюємо порядок сортування на зростання

dataGridView1.Sort(dataGridView1.Columns[dataGridView1.CurrentCell.ColumnIndex
], System.ComponentModel.ListSortDirection.Ascending);
                }
```

```
            }
            if (dataGridView2.Focused)
            {
                if (dataGridView2.SortOrder == SortOrder.Ascending)
                {
                    // Якщо DataGridView вже сортується за вибраною колонкою в
порядку зростання,
                    // змінюємо порядок сортування на спадання

dataGridView2.Sort(dataGridView2.Columns[dataGridView2.CurrentCell.ColumnIndex
], System.ComponentModel.ListSortDirection.Descending);
                }
                else
                {
                    // Якщо DataGridView сортується за вибраною колонкою в
порядку спадання або не сортується взагалі,
                    // змінюємо порядок сортування на зростання

dataGridView2.Sort(dataGridView2.Columns[dataGridView2.CurrentCell.ColumnIndex
], System.ComponentModel.ListSortDirection.Ascending);
                }
            }
        }

        private void сортуванняЗаРядкамиToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            if (dataGridView1.Focused)
            {
                // Отримуємо індекс виділеного рядка
                int selectedRowIndex =
dataGridView1.SelectedCells[0].RowIndex;

                // Отримуємо значення виділеного рядка
                DataGridViewRow selectedRow =
dataGridView1.Rows[selectedRowIndex];

                // Отримуємо значення з виділеного рядка
                List<object> rowValues = new List<object>();

                foreach (DataGridViewCell cell in selectedRow.Cells)
                {
                    rowValues.Add(cell.Value);
                }

                // Сортуємо значення
                rowValues.Sort();

                // Записуємо відсортовані значення у виділений рядок
                for (int i = 0; i < rowValues.Count; i++)
                {
                    selectedRow.Cells[i].Value = rowValues[i];
                }

                // Оновлюємо DataGridView, щоб показати зміни
                dataGridView1.Refresh();
            }
            else if (dataGridView2.Focused)
            {
                // Отримуємо індекс виділеного рядка
                int selectedRowIndex =
dataGridView2.SelectedCells[0].RowIndex;

                // Отримуємо значення виділеного рядка
```

```
                DataGridViewRow selectedRow =
dataGridView2.Rows[selectedRowIndex];

                // Отримуємо значення з виділеного рядка
                List<object> rowValues = new List<object>();

                foreach (DataGridViewCell cell in selectedRow.Cells)
                {
                    rowValues.Add(cell.Value);
                }

                // Сортуємо значення
                rowValues.Sort();

                // Записуємо відсортовані значення у виділений рядок
                for (int i = 0; i < rowValues.Count; i++)
                {
                    selectedRow.Cells[i].Value = rowValues[i];
                }

                // Оновлюємо DataGridView, щоб показати зміни
                dataGridView2.Refresh();
            }
        }

        private void дозволитиРедагуватиToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            if (dataGridView1.Focused)
            {
                foreach (DataGridViewColumn column in dataGridView1.Columns)
                {
                    column.ReadOnly = false;
                }
            }
            else if (dataGridView2.Focused)
            {
                foreach (DataGridViewColumn column in dataGridView2.Columns)
                {
                    column.ReadOnly = false;
                }
            }
        }

        private void заборонитиРедагуватиToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            if (dataGridView1.Focused)
            {
                foreach (DataGridViewColumn column in dataGridView1.Columns)
                {
                    column.ReadOnly = true;
                }
            }
            else if (dataGridView2.Focused)
            {
                foreach (DataGridViewColumn column in dataGridView2.Columns)
                {
                    column.ReadOnly = true;
                }
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
```

```
            // Отримати кількість рядків і стовпців у DataGridView1 та
DataGridView2
            int rowsCount = dataGridView1.Rows.Count;
            int columnsCount = dataGridView1.Columns.Count;

            // Перевірка, чи кількість рядків і стовпців у DataGridView1 і
DataGridView2 однакова
            if (dataGridView2.Rows.Count != rowsCount ||
dataGridView2.Columns.Count != columnsCount)
            {
                MessageBox.Show("Кількість рядків і стовпців у DataGridView1 і
DataGridView2 повинна бути однаковою.");
                return;
            }

            // Створення DataGridView3 з відповідними розмірами
            dataGridView3.RowCount = rowsCount;
            dataGridView3.ColumnCount = columnsCount;

            // Обчислення суми значень в клітинках DataGridView1 і
DataGridView2 і запис в DataGridView3
            for (int row = 0; row < rowsCount; row++)
            {
                for (int column = 0; column < columnsCount; column++)
                {
                    if (dataGridView1.Rows[row].Cells[column].Value == null &&
dataGridView2.Rows[row].Cells[column].Value == null)
                        return;
                    // Отримати значення з клітинок DataGridView1 і
DataGridView2
                    int value1 =
Convert.ToInt32(dataGridView1.Rows[row].Cells[column].Value);
                    int value2 =
Convert.ToInt32(dataGridView2.Rows[row].Cells[column].Value);

                    // Обчислити суму значень
                    int sum = value1 + value2;

                    // Записати суму в клітинку DataGridView3
                    dataGridView3.Rows[row].Cells[column].Value = sum;
                }
            }
        }

        private void button2_Click(object sender, EventArgs e)
        {
            // Отримати кількість рядків і стовпців у DataGridView1
            int rowsCount = dataGridView1.Rows.Count;
            int columnsCount = dataGridView1.Columns.Count;

            // Створити DataGridView3 з відповідними розмірами
            dataGridView3.RowCount = columnsCount;
            dataGridView3.ColumnCount = rowsCount;

            // Транспонування матриці
            for (int row = 0; row < rowsCount; row++)
            {
                for (int column = 0; column < columnsCount; column++)
                {
                    // Отримати значення з клітинки DataGridView1
                    object value =
dataGridView1.Rows[row].Cells[column].Value;

                    // Записати значення в відповідну клітинку DataGridView3
                    dataGridView3.Rows[column].Cells[row].Value = value;
```

```
                }
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            // Отримати кількість рядків і стовпців у DataGridView1 і
DataGridView2
            int rowsCount = dataGridView1.Rows.Count;
            int columnsCount = dataGridView1.Columns.Count;

            // Перевірка, чи кількість рядків і стовпців у DataGridView1 і
DataGridView2 однакова
            if (dataGridView2.Rows.Count != rowsCount ||
dataGridView2.Columns.Count != columnsCount)
            {
                MessageBox.Show("Кількість рядків і стовпців у DataGridView1 і
DataGridView2 повинна бути однаковою.");
                return;
            }

            // Створити DataGridView3 з відповідними розмірами
            dataGridView3.RowCount = rowsCount;
            dataGridView3.ColumnCount = columnsCount;

            // Порівняти числа з DataGridView1 і DataGridView2 та записати
збігаючіся числа в DataGridView3
            for (int row = 0; row < rowsCount; row++)
            {
                for (int column = 0; column < columnsCount; column++)
                {
                    if (dataGridView1.Rows[row].Cells[column].Value == null &&
dataGridView2.Rows[row].Cells[column].Value == null)
                        return;
                    // Отримати значення з клітинок DataGridView1 і
DataGridView2
                    int value1 =
Convert.ToInt32(dataGridView1.Rows[row].Cells[column].Value);
                    int value2 =
Convert.ToInt32(dataGridView2.Rows[row].Cells[column].Value);

                    // Порівняти числа
                    if (value1 == value2)
                    {
                        // Записати збігаючіся числа в клітинку DataGridView3
                        dataGridView3.Rows[row].Cells[column].Value = value1;
                    }
                    else
                    {
                        // Залишити порожню клітинку в DataGridView3
                        dataGridView3.Rows[row].Cells[column].Value = null;
                    }
                }
            }
        }

        private void проПрограмуToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            AboutBox1 aboutBox = new AboutBox1();
            aboutBox.Show();
        }

        private void завершитиРоботуToolStripMenuItem_Click(object sender,
EventArgs e)
```

```
        {
            Close();
        }
    }
}
```

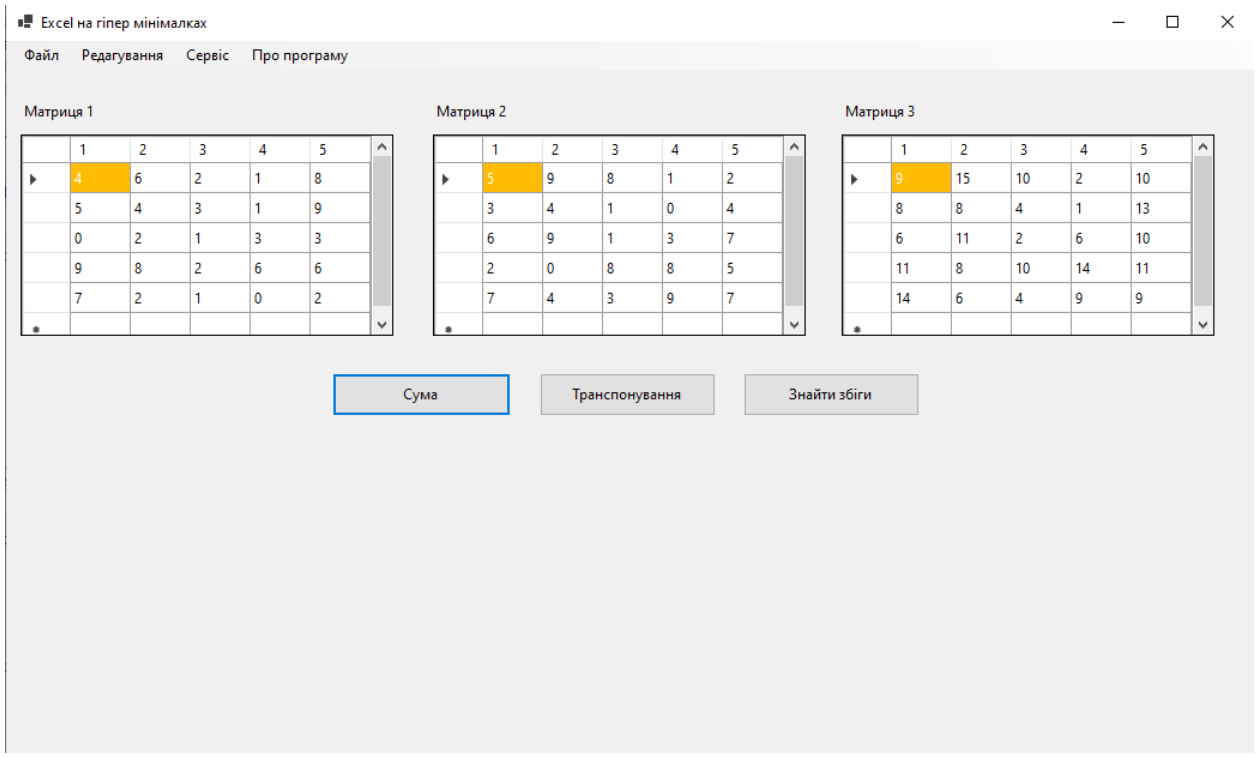
AboutBox1.cs

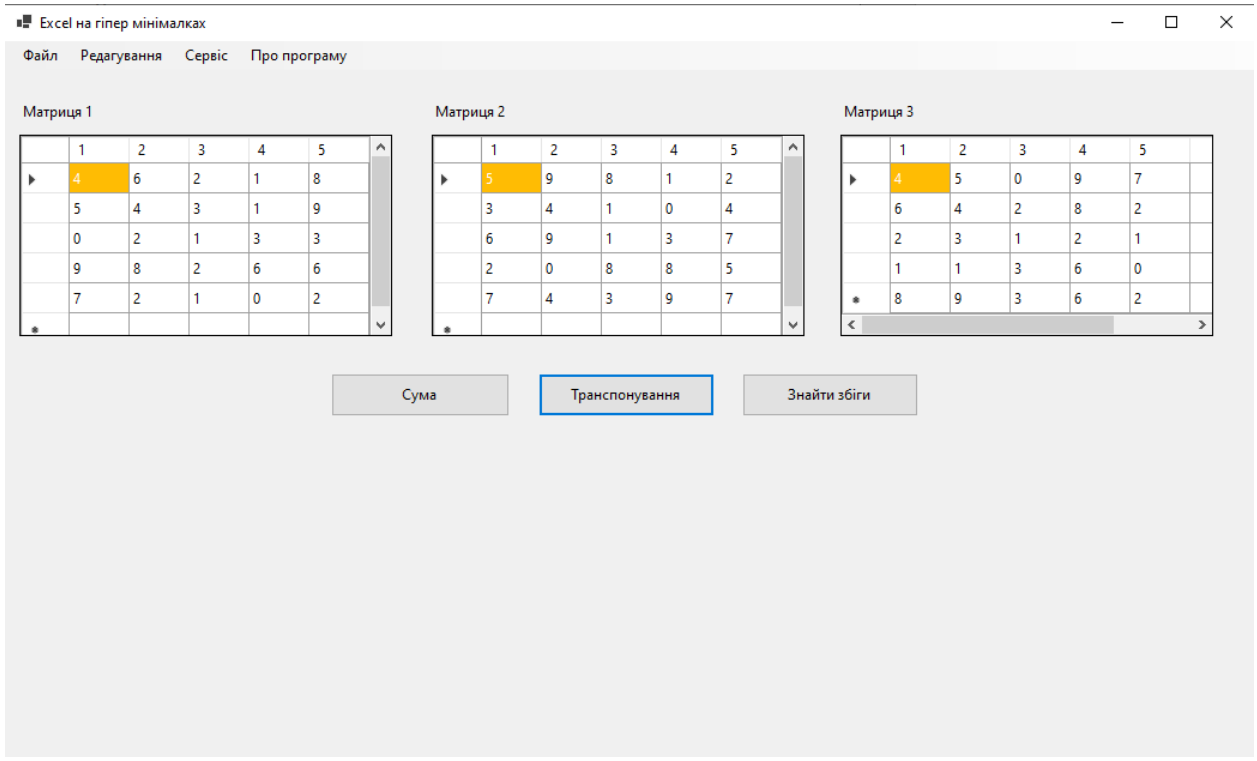
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Drawing;
using System.Linq;
using System.Reflection;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab3
{
    partial class AboutBox1 : Form
    {
        public AboutBox1()
        {
            InitializeComponent();
            this.Text = "Про програму";
            this.labelProductName.Text = "Excel на гіпер мінімалках";
            this.labelVersion.Text = String.Format("Version {0}",
AssemblyVersion);
            this.labelCopyright.Text = AssemblyCopyright;
            this.labelCompanyName.Text = "Rostik Hubariev";
        }

        #region Assembly Attribute Accessors

        public string AssemblyTitle
        {
            get
            {
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttrib
ute), false);
                if (attributes.Length > 0)
                {
                    AssemblyTitleAttribute titleAttribute =
(AssemblyTitleAttribute)attributes[0];
                    if (titleAttribute.Title != "")
                    {
                        return titleAttribute.Title;
                    }
                }
                return
System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().Cod
eBase);
            }
        }

        public string AssemblyVersion
        {
            get
            {
                return
Assembly.GetExecutingAssembly().GetName().Version.ToString();
            }
        }

        public string AssemblyDescription
        {
            get
            {
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescription
Attribute), false);
```

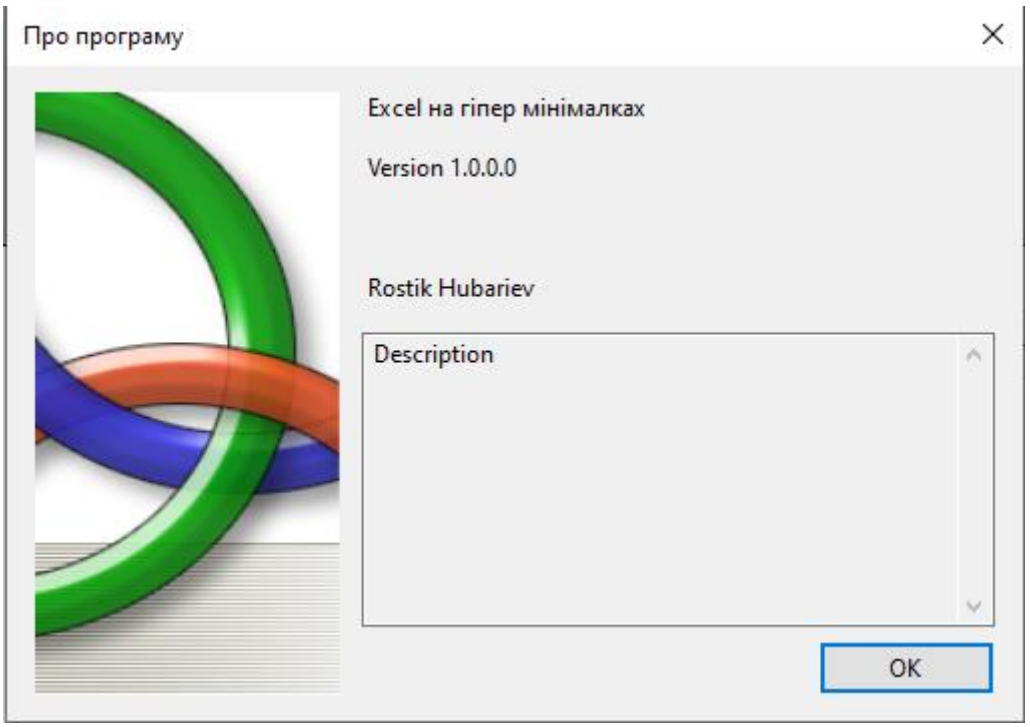
```
                if (attributes.Length == 0)
                {
                    return "";
                }
                return
((AssemblyDescriptionAttribute)attributes[0]).Description;
            }
        }

        public string AssemblyProduct
        {
            get
            {
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttr
ibute), false);
                if (attributes.Length == 0)
                {
                    return "";
                }
                return ((AssemblyProductAttribute)attributes[0]).Product;
            }
        }

        public string AssemblyCopyright
        {
            get
            {
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAt
tribute), false);
                if (attributes.Length == 0)
                {
                    return "";
                }
                return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;
            }
        }

        public string AssemblyCompany
        {
            get
            {
                object[] attributes =
Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttr
ibute), false);
                if (attributes.Length == 0)
                {
                    return "";
                }
                return ((AssemblyCompanyAttribute)attributes[0]).Company;
            }
        }
        #endregion

        private void AboutBox1_Load(object sender, EventArgs e)
        {

        }

        private void okButton_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
```

```
}
```

4. Знімок інтерфейсу програми з результатами роботи







5. Короткі висновки

В цій лабораторній роботі я навчився працювати з об’єктом DataGridView та його властивостями.

6. Список літератури

- **https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.datagridview?view=windowsdesktop-7.0**