ЛАБОРАТОРНА РОБОТА № 3

Тема: Багатопотоковість у C#.
Мета роботи: навчитися працювати з потоками та процесами у мові C#.

Хід роботи:

В. ***Розробити дві програми*:**
1) шифрування файлів;
2) менеджер процесів.
Програмний код має бути написаний максимально універсально без прив'язки у класах, що реалізують основний функціонал до інтерфейсу. Передбачається, що дані класи потрібно буде використовувати в наступних лабораторних роботах.

Вимоги до програми шифрування файлів

1. Графічний інтерфейс користувача з можливістю вводу ключа для шифрування та діалогу вибору довільного файлу, який буде шифруватись або розшифровуватись.

2. Здатність працювати з файлами довільного розміру та формату.

3. В процесі шифрування, повинен відображатись індикатор прогресу (0-100%) та час, який пройшов від запуску шифрування.

4. По завершенні шифрування повинне бути відображене вікно з інформацією про розмір зашифрованого файла, його назву та час, затрачений на шифрування.

5. Необхідно запобігти «підвисанню» вікна при здійсненні операції шифрування (шифрування здійснюватись в окремому обчислювальному потоці за допомогою класу BackgroundWorker) .

6. Коректна обробка виключень, що можуть виникати під час роботи.

7. Для успішного виконання роботи необхідно ознайомитися з наступними темами: асинхронні операції (клас BackgroundWorker), класи File, FileStream, Timer, DateTime, OpenFileDialog, SaveFileDialog, обробка виключень.

Лістинг програми form1:

```
using System.Diagnostics;

namespace Encryptor
{
```

```
 public partial class Form1 : Form
    {
        public string FileContent { get; set; } = string.Empty;
        public string FileName { get; set; }
        public Form1()
        {

InitializeComponent();
            backgroundWorker1.WorkerReportsProgress = true;
            resultTextBox.Enabled = false;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            backgroundWorker1.RunWorkerAsync("Encrypt");
        }

        private void backgroundWorker1_DoWork(object sender,
System.ComponentModel.DoWorkEventArgs e)
        {
            var timer = new Stopwatch();

            timer.Start();

            var result = FileEncryptor.Encrypt(FileContent, textBox1.Text,
backgroundWorker1);

            timer.Stop();

            TimeSpan timeTaken = timer.Elapsed;

            string foo = "Time taken: " + timeTaken.ToString(@"m\:ss\.fff");

            timeTakenLabel.Text = foo;

            e.Result = result;
        }

        private void backgroundWorker1_ProgressChanged(object sender,
System.ComponentModel.ProgressChangedEventArgs e)
        {
            progressBar1.Value = e.ProgressPercentage;
        }

        private void backgroundWorker1_RunWorkerCompleted(object sender,
System.ComponentModel.RunWorkerCompletedEventArgs e)
        {
            resultTextBox.Text = e.Result.ToString();
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
            DialogResult result = openFileDialog1.ShowDialog();

            if (result == DialogResult.OK)
            {
                string sourcePath = openFileDialog1.FileName;

                FileContent = File.ReadAllText(sourcePath);
            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {
```

```
        }
    }
}
```

Лістинг програми FileEncryptor:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Encryptor
{
    public static class FileEncryptor
    {
        private static string XORCipher(string inputContent, string key, BackgroundWorker
backgroundWorker)
        {
            if (inputContent == null || key == null)
                throw new ArgumentNullException("Wrong argument passed to file
encryptor");

            int dataLen = inputContent.Length;
            int keyLen = key.Length;

            char[] output = new char[dataLen];

            for (int i = 0; i < dataLen; ++i)
            {
                output[i] = (char)(inputContent[i] ^ key[i % keyLen]);

                int progressPercent = (int)(((i + 1.0) / inputContent.Length) * 100);
                backgroundWorker.ReportProgress(progressPercent);
            }

            return new string(output);
        }

        public static string Encrypt(string inputContent, string key, BackgroundWorker
backgroundWorker) =>
            XORCipher(inputContent, key, backgroundWorker);

        public static string Decrypt(string inputContent, string key, BackgroundWorker
backgroundWorker) =>
            XORCipher(inputContent, key, backgroundWorker);
    }
}
```

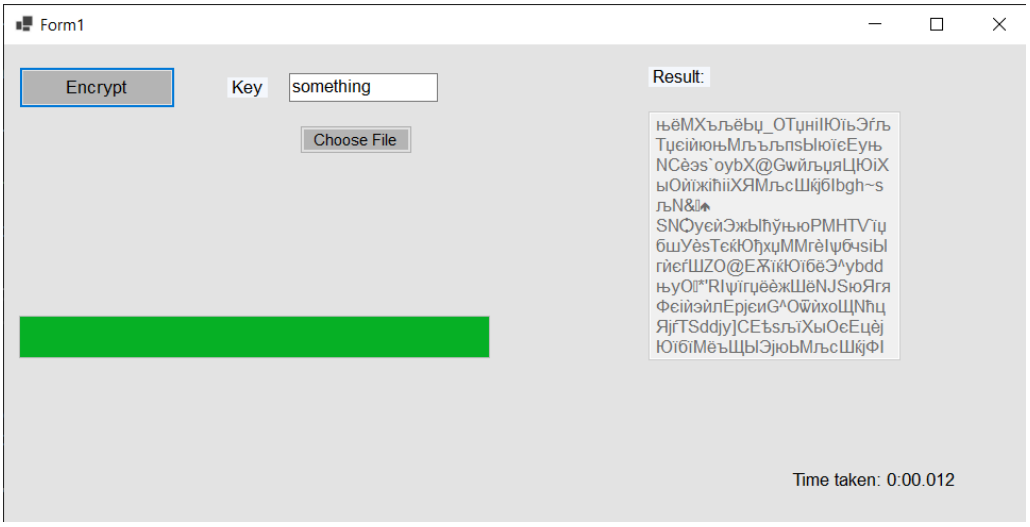

Рис 1. Результат виконання програми

Вимоги до менеджера процесів

1. Можливість запуску програм: Калькулятор, Microsoft Word та ще трьох на ваш вибір.

2. Перегляд список запущених процесів з інформацією про обсяг оперативної пам'яті, виділений під процес, часу, коли був запущений процес, пріоритет процесу, кількість потоків, які запущені процесом. Відобразити у вигляді таблиці.

3. Можливість зупинки вибраного процесу.

4. Можливість зміни пріоритету вибраного процесу.

Лістинг програми Form:

```
using System.Diagnostics;
using System.Text;

namespace ProcessManager
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            GetProcessesInfo();

            comboBox1.DataSource = new ComboItem[]
            {
                new ComboItem { ID = 1, Text = "Lowest", Value =
ThreadPriorityLevel.Lowest},
                new ComboItem { ID = 2, Text = "Normal", Value =
ThreadPriorityLevel.Normal},
                new ComboItem { ID = 3, Text = "Idle", Value = ThreadPriorityLevel.Idle},
                new ComboItem { ID = 4, Text = "Highest", Value =
ThreadPriorityLevel.Highest}
            };
        }
        public void GetProcessesInfo()
        {
```

```
            dataGridView1.ColumnCount = 4;

            dataGridView1.Columns[0].Name = "ProcessName";
            dataGridView1.Columns[1].Name = "MemoryTaken";
            dataGridView1.Columns[2].Name = "ThreadsCount";
            dataGridView1.Columns[3].Name = "ThreadPriority";

            var rows = ProcessUtilities.GetProcessesInfo();

            foreach (var item in rows)
            {
                dataGridView1.Rows.Add(item);
            }
        }

        private void runChromeButton_Click(object sender, EventArgs e)
        {
            ProcessUtilities.StartProcessByPathName(@"C:\Program
Files\Google\Chrome\Application\chrome.exe");
        }

        private void runVSCodeButton_Click(object sender, EventArgs e)
        {
            ProcessUtilities.StartProcessByPathName(@"C:\Program Files\Microsoft VS
Code\Code.exe");
        }

        private void stopProcessButton_Click(object sender, EventArgs e)
        {
            Int32 selectedRowCount =
                 dataGridView1.Rows.GetRowCount(DataGridViewElementStates.Selected);

            if (selectedRowCount > 0)
            {

                var processName = (string)dataGridView1.SelectedRows[0].Cells[0].Value;

                ProcessUtilities.StopProcessByName(processName);
            }
        }

        private void setPriorityOfChoosenThreadButton_Click(object sender, EventArgs e)
        {
            Int32 selectedRowCount =
                dataGridView1.Rows.GetRowCount(DataGridViewElementStates.Selected);

           if (selectedRowCount > 0)
           {
                ComboItem value = (ComboItem)comboBox1.SelectedItem;

                var processName = (string)dataGridView1.SelectedRows[0].Cells[0].Value;

                var chosenProcess = Process.GetProcessesByName(processName);

                foreach (var process in chosenProcess)
                {
                    foreach (ProcessThread thread in process.Threads)
                    {
                        thread.PriorityLevel = value.Value;
                    }
                }

                GetProcessesInfo();
            }
        }
```

```
        private void runCalcButton_Click(object sender, EventArgs e)
        {
            ProcessUtilities.StartProcessByPathName("Calc.exe");
        }

        private void runMSWordButton_Click(object sender, EventArgs e)
        {
            ProcessUtilities.StartProcessByPathName("Winword.exe");
        }
    }
    class ComboItem
    {
        public int ID { get; set; }
        public string Text { get; set; }
        public ThreadPriorityLevel Value { get; set; }
    }
      }
```

Лістинг програми processUtilities:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace ProcessManager
{
    public static class ProcessUtilities
    {
        public static void StartProcessByPathName(string pathName)
        {
            Process.Start(pathName);
        }

        public static void StopProcessByName(string name)
        {
            var chosenProcess = Process.GetProcessesByName(name);

            foreach (var process in chosenProcess)
            {
                process.Kill();
            }
        }

        public static string[][] GetProcessesInfo()
        {
            List<string[]> result = new();

            foreach (var process in Process.GetProcesses())
            {
                var name = process.ProcessName;
                var memoryTaken = process.PagedMemorySize64;
                var threadsCount = process.Threads.Count;

                StringBuilder sb = new StringBuilder();

                foreach (ProcessThread thread in process.Threads)
                {
```

```
                sb.Append($" {thread.CurrentPriority}");
            }

            string[] row = new string[] { name, memoryTaken.ToString(),
threadsCount.ToString(), sb.ToString() };

            result.Add(row);
        }

        return result.ToArray();
    }
}
```

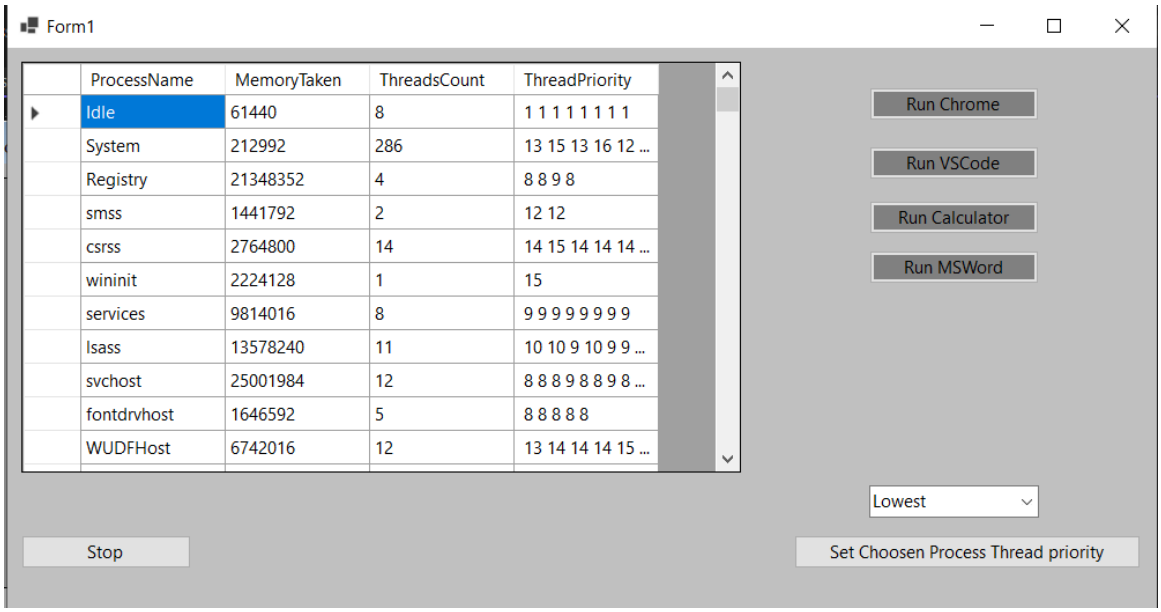

Рис 2. Результат виконання програми

Висновок: в ході виконання лабораторної роботи я навчилася працювати з потоками та процесами у мові C#.

Посилання на GitHub: https://github.com/Vika-Myronova/DotNetLab03