Міністерство освіти і науки України

Державний університет „Житомирська політехніка”

Кафедра ІПЗ

Група: ІПЗ-21-2

Технології розробки додатків .NET Core

Лабораторна робота №3

«Багатопотоковість у C#»

Виконав: Троцюк Ю.М.

Прийняв: Чижмотря О.В.

					22.121.26.000 – Лр3			
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розроб.		*Троцюк Ю.М.*			Звіт з лабораторної роботи	*Літ.*	*Арк.*	*Аркушів*
Перевір.		*ЧижмотряО.В*					*1*	*15*
Керівник						*ФІКТ, гр. ІПЗ-21-2*		
Н. контр.								
Затверд.								

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

Виконання роботи:

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.

1. ШИФРУВАННЯ ФАЙЛІВ.

Лістинг класу Crypt:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO.Compression;
using System.ComponentModel;
using System.IO;
using CodeLibrary;
using System.Diagnostics;


namespace ArchivatorLibrary
{
    public class Crypt
    {

        public string Filepath { private set; get; }
        public string Password { private set; get; }
```

```
        public Crypt(string path,string pass)
        {

            Filepath = path;
            Password = pass;
        }

        public Crypt()
        {
        }

        public float FileSize()
        {
            FileInfo file = new FileInfo(Filepath);
            return file.Length;
        }

        public event EventHandler<EventHandlerProgres> Progress;
        protected void InProgress(int percent)
        {
            if (Progress != null)
                Progress(this, new EventHandlerProgres(percent));
        }

        public event EventHandler<EventHandlerCodeTime> Timer;
        protected void InTime(int strHour, int strMin, int strSec, int milisec)
        {
            if (Timer != null)
                Timer(this, new EventHandlerCodeTime(strHour, strMin, strSec, milisec));
        }


        public void CryptFile()
        {
            Stopwatch watch = new Stopwatch();
            BackgroundWorker worker = new BackgroundWorker();
            worker.DoWork += (o, e) =>
                {
                    using (FileStream flstrIN = File.OpenRead(Filepath))
                    using (FileStream flstrOUT = File.OpenWrite(Filepath + ".coded"))
                    {
                        watch.Start();
                        int symbol;
                        int counter = 1;
                        int PasPosition = 0;
                        while ((symbol = flstrIN.ReadByte()) != -1)
                        {
                            symbol = symbol ^ Password[PasPosition];
                            PasPosition++;
                            flstrOUT.WriteByte((byte)symbol);
                            if (PasPosition >= Password.Length) PasPosition = 0;

                            InProgress((int)((counter * 1.0) / flstrIN.Length * 100));
                            counter++;
                        }

                    }
                    watch.Stop();
                    InTime(watch.Elapsed.Hours, watch.Elapsed.Minutes,
watch.Elapsed.Seconds, watch.Elapsed.Milliseconds);

                };
            worker.RunWorkerAsync();
        }
```

```
        public void DecriptFile()
        {
            BackgroundWorker worker = new BackgroundWorker();
            worker.DoWork += (o, e) =>
            {
                using (FileStream flstrIN = File.OpenRead(Filepath))
                {
                    Filepath = Filepath.Remove(Filepath.Length - 6);
                    using (FileStream flstrOUT = File.OpenWrite(Filepath + ".decoded"))
                    {
                        int symbol;
                        int counter = 1;
                        int PasPosition = 0;
                        while ((symbol = flstrIN.ReadByte()) != -1)
                        {
                            symbol = symbol ^ Password[PasPosition];
                            PasPosition++;
                            flstrOUT.WriteByte((byte)symbol);
                            if (PasPosition >= Password.Length) PasPosition = 0;

                            InProgress((int)((counter * 1.0) / flstrIN.Length * 100));
                            counter++;
                        }
                    }
                }
            };
            worker.RunWorkerAsync();
        }
    }
}
namespace DotNetLab2
{
    public static class StringExtensions
    {
        //Інвертування рядка
        public static string? InverToString(this string? str)
        {
            string? strNew = null;

            if (str != null)
                for (int i = str.Length - 1; i >= 0; i--)
                    strNew += str[i];
            return strNew;
        }

        //Підрахунок кількості входжень заданого у параметрі символа у рядок
        public static int CharCount(this string? str, char ch)
        {
            int count = 0;
            if (str != null)
                for (int i = 0; i < str.Length; i++)
                    if (ch == str[i])
                        count++;
            return count;
        }
    }
}
```

Лістинг класу EventHandlerCodeTime:

```
using System;
using System.Collections.Generic;
using System.Text;
```

```
namespace CodeLibrary
{
    public class EventHandlerCodeTime:EventArgs
    {
        public float timeSeconds { set; get; }
        public int timeMinutes { set; get; }
        public int timeHours { set; get; }
        public int timeMilisec { set; get; }

        public string totalTimeString { set; get; }

        public EventHandlerCodeTime(int strHour, int strMin, int strSec, int milisec)
        {
            timeHours = strHour;
            timeMinutes = strMin;
            timeSeconds = strSec;
            timeMilisec = milisec;

            if (timeHours == 0 && timeMinutes != 0)
            {
                totalTimeString = $"{timeMinutes}:{timeSeconds} sec.";
            }
            else if (timeHours == 0 && timeMinutes == 0)
            {
                totalTimeString = $"{timeSeconds}.{timeMilisec} sec.";
            }

        }
    }
}
```

Лістинг класу Program:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using ArchivatorLibrary;
using CodeLibrary;

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


        }
        private string file;
        private float filesize;
        private void openToolStripMenuItem_Click(object sender, EventArgs e)
        {
            OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.Filter = "всі файли (*.*)|*.*";
```

```
        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            file = openFileDialog.FileName;
            removeToolStripMenuItem.Visible = true;
            decodeToolStripMenuItem.Visible = true;
            codeToolStripMenuItem.Visible = true;
            labelProgress.Visible = false;
            panelTimer.Visible = false;
            labelHome.Text = "Виберіть дію";

        }

    }


    private bool ErrorExceptionPass()
    {
        bool flag = true;
        if (textBoxPass.Text.Length == 0)
        {
            MessageBox.Show("Password can't be a null");
            flag = false;
        }
        return flag;
    }

    private void ProgressBarUpdateAction(object sender, EventHandlerProgres e)
    {
        this.Invoke(
            new Action(() =>
            {
                if (e.progress == 100)
                {
                    labelHome.Text = "Успішно!";
                    labelHome.Visible = true;
                    progressBar.Visible = false;
                    openToolStripMenuItem.Visible = true;
                }
                progressBar.Value = e.progress;
                labelProgress.Text = e.progress.ToString() + "%";
    }));
    }

    private void Timer(object sender, EventHandlerCodeTime e)
    {
        this.Invoke(
            new Action(() =>
            {
                panelTimer.Visible = true;
                labelTimer.Text = e.totalTimeString;
                labelSizeFile.Text = filesize.ToString()+"Bites";
            }));
    }

    int flag;
    private void codeToolStripMenuItem_Click(object sender, EventArgs e)
    {
        labelHome.Text = "Шифрування";
        panelPass.Visible = true;
        decodeToolStripMenuItem.Visible = false;
        flag = 1;
    }
```

```
        private void decodeToolStripMenuItem_Click(object sender, EventArgs e)
        {
            labelHome.Text = "Розшифрування";
            panelPass.Visible = true;
            flag = 0;
            codeToolStripMenuItem.Visible = false;
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            panelPass.Visible = false;
        }

        private void buttonPass_Click(object sender, EventArgs e)
        {
            if (!ErrorExceptionPass())
                return;

            Crypt crypt = new Crypt(file, textBoxPass.Text);
            crypt.Progress += ProgressBarUpdateAction;
            crypt.Timer += Timer;
            filesize = (float)(crypt.FileSize());
            labelProgress.Visible = true;

            if (flag == 1)
               crypt.CryptFile();
            if (flag == 0)
               crypt.DecriptFile();


            panelPass.Visible = false;
            removeToolStripMenuItem.Visible = false;
            decodeToolStripMenuItem.Visible = false;
            codeToolStripMenuItem.Visible = false;
            openToolStripMenuItem.Visible = false;

        }
    }
}
```

Результат виконання:

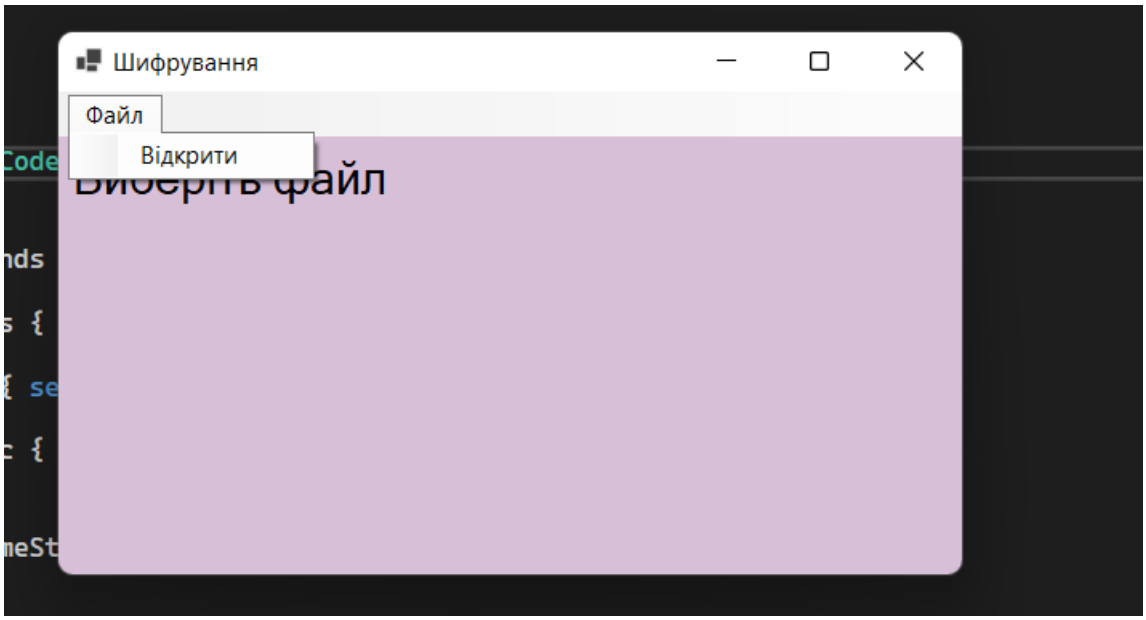


Рис.1 – Демонстрація запуску програми для шифрування

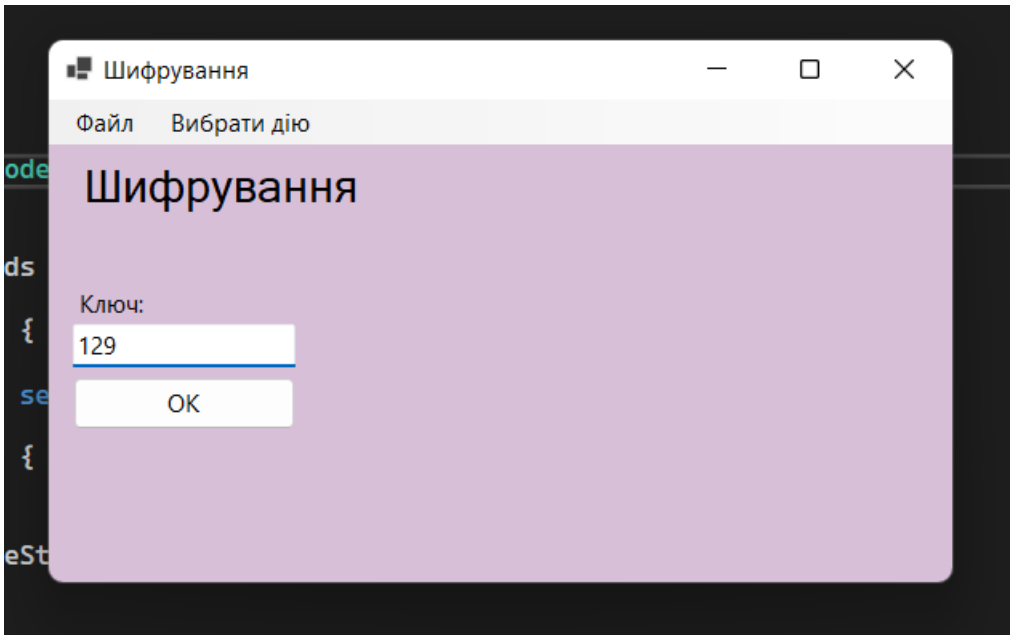


Рис.2 – Демонстрація шифрування вибраного файлу

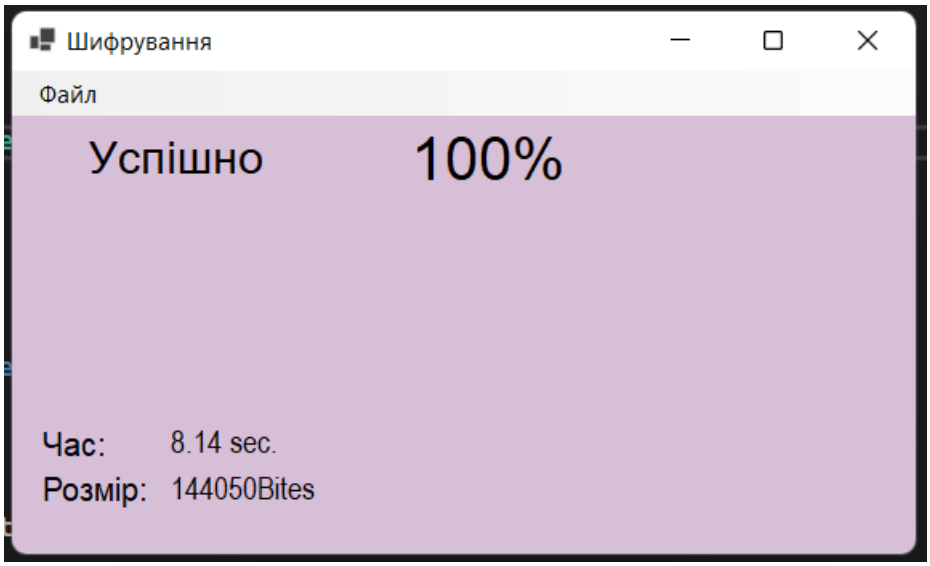


Рис.3 – Демонстрація успішного шифрування

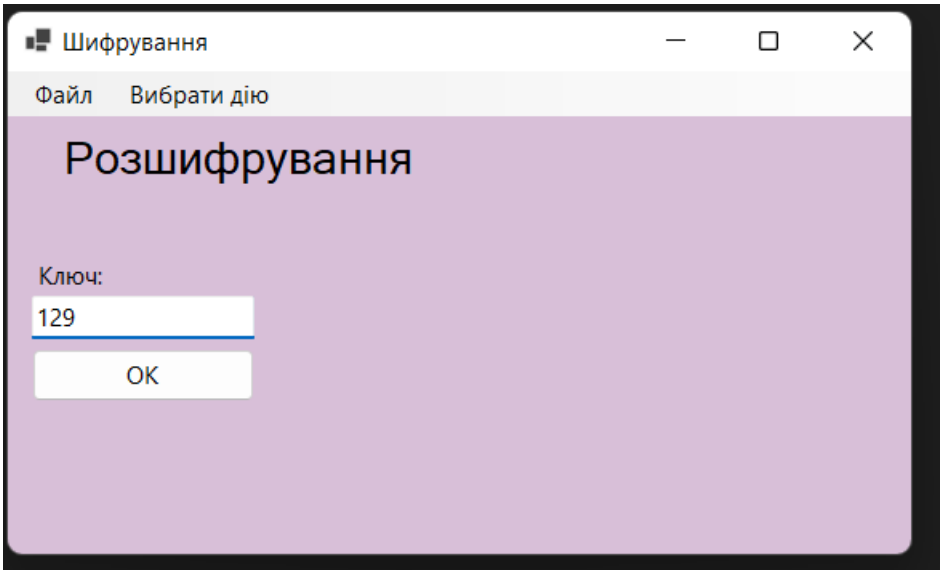


Рис.4 – Демонстрація розшифрування вибраного файлу

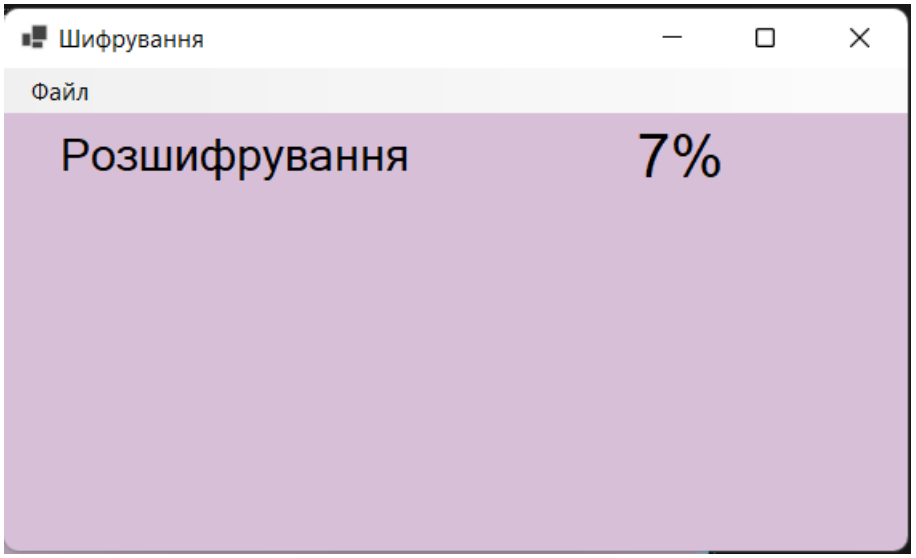


Рис.5 – Демонстрація прогресу розшифрування

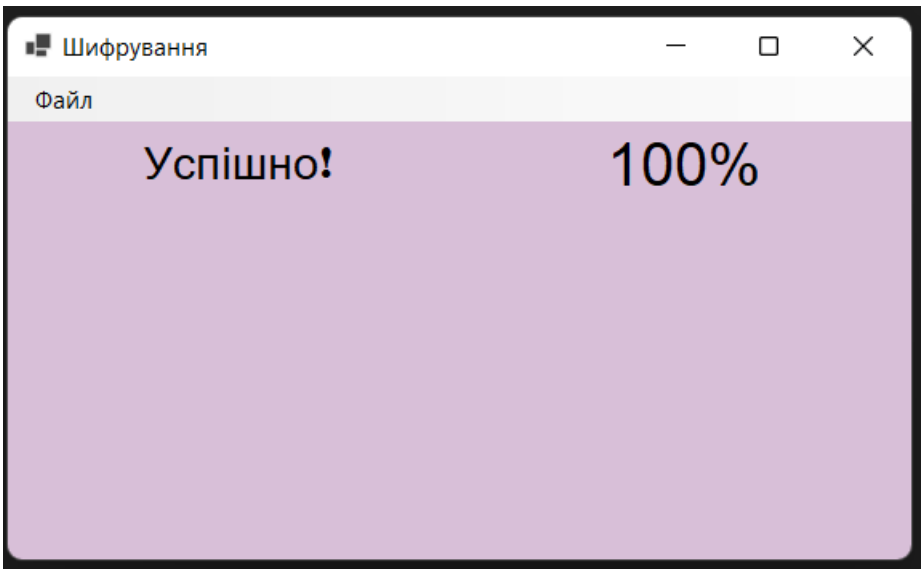


Рис.6 – Демонстрація успішного шифрування

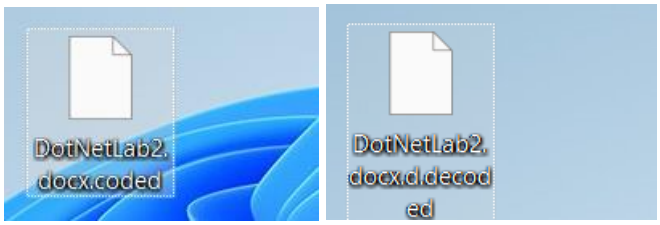


Рис.7 – Демонстрація успішного виконання програми

2. МЕНЕДЖЕР ПРОЦЕСІВ.

Лістинг класу Program:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Reflection.Metadata;

namespace ProcessApplication
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private int RowPosition;
        private Process[] processes;
```

```
        private int ProcessDataGridId;
        private void processesToolStripMenuItem_Click(object sender, EventArgs e)
        {

            panelInfo.Visible = false;

        }

        private void processesInfoToolStripMenuItem_Click(object sender, EventArgs e)
        {
            panelStart.Visible = false;
            dataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
            panelInfo.Visible = true;
            this.Width = 818;
            this.Height = 511;
                GetProcesses();

        }

        private void GetProcesses()
        {
            processes = Process.GetProcesses();
            dataGridView1.RowCount = processes.Length;
            for (int i = 0; i < dataGridView1.RowCount; i++)
            {
                double mem = processes[i].PagedMemorySize64 / 1048576.0;
                dataGridView1.Rows[i].HeaderCell.Value = i.ToString();
                dataGridView1[0, i].Value = processes[i].ProcessName;
                dataGridView1[1, i].Value = $"{mem:F1}";
                //dataGridView1[2, i].Value = processes[i].StartTime; ///no access
                dataGridView1[3, i].Value = ReturnPriorityType(processes[i]);
                dataGridView1[4, i].Value = processes[i].Threads.Count;

            }
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            this.Width = 694;
            this.Height = 164;
            dataGridView1.RowHeadersWidth = 65;
            dataGridView1.MouseClick += new MouseEventHandler(DataGrid_MouseClick);
        }

        private void DataGrid_MouseClick(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Right)
            {


                ContextMenuStrip myMenu = new ContextMenuStrip();
                RowPosition = dataGridView1.HitTest(e.X, e.Y).RowIndex;
                string StrProcessDataGridId =
dataGridView1.Rows[RowPosition].HeaderCell.Value.ToString();
                //коли відбувається впорядковування таблиці по значенню стовпця, звичайна
позиція RowPosition не може бути використана.
                ProcessDataGridId = int.Parse(StrProcessDataGridId);
                dataGridView1.ClearSelection();
                dataGridView1.Rows[RowPosition].Selected = true;


                if (ProcessDataGridId >= 0)
                {
```

```
                    myMenu.Items.Add("Kill Process").Name = "Kill Process";
                    myMenu.Items.Add("Priority Normal").Name = "Priority Normal";
                    myMenu.Items.Add("Priority High").Name = "Priority High";

                }

                myMenu.Show(dataGridView1, new Point(e.X, e.Y));
                myMenu.ItemClicked += new
ToolStripItemClickedEventHandler(myMenu_ItemClicked);
            }
        }

        private void myMenu_ItemClicked(object sender, ToolStripItemClickedEventArgs e)
        {

            switch (e.ClickedItem.Name.ToString())
            {
                case "Kill Process":
                    MessageBox.Show(e.ClickedItem.Name.ToString());
                    MessageBox.Show(ProcessDataGridId.ToString());
                    MessageBox.Show(processes[ProcessDataGridId].ProcessName.ToString());
                    processes[ProcessDataGridId].Kill();
                    GetProcesses();
                    break;

                case "Priority Normal":
                    processes[ProcessDataGridId].PriorityClass =
ProcessPriorityClass.Normal;
                    GetProcesses();

                    break;

                case "Priority High":
                    processes[ProcessDataGridId].PriorityClass =
ProcessPriorityClass.High;
                    GetProcesses();

                    break;
            }
        }


        private string ReturnPriorityType(Process process)
        {
            Dictionary<int, string> prior = new Dictionary<int, string>()
            {
                 {0, "0"},
                {4, "Idle"},
                 {5, "Idle"},
                  {6, "Idle"},
                   {7, "Idle"},
                  {8, "Normal"},
                    {9, "Normal"},
                {10, "Normal"},
                {11, "Normal"},
               {12, "Normal"},
              {13, "High"},
                {24, "RealTime"}

            };
            string str = process.BasePriority.ToString();
            return prior[Int32.Parse(str)];

        }
```

```
        private void calculatorToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Process.Start("calc");
            labelStart.Text = "Calculator start";
        }

        private void microsoftWordToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Process.Start(@"C:\Program Files\Microsoft
Office\root\Office16\WINWORD.EXE");
            labelStart.Text = "Microsoft Word started";
        }

        private void microsoftExelToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Process.Start(@"C:\Program Files\Microsoft Office\root\Office16\EXCEL.EXE");
            labelStart.Text = "Microsoft Exel started";

        }

        private void microsoftPowerToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Process.Start(@"C:\Program Files\Microsoft
Office\root\Office16\POWERPNT.EXE");
            labelStart.Text = "Microsoft PowerPoint started";

        }

        private void microsoftAccessToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Process.Start(@"C:\Program Files\Microsoft
Office\root\Office16\MSACCESS.EXE");
            labelStart.Text = "Microsoft Access started";
        }

        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }
    }
}
```

Результат виконання:

Рис.2 – Демонстрація методів розширення для одновимірного масиву

Також ці методи розширення працюють і для інших типів даних:

```
string[] array = {"bool","int","string","int","int","char"};
Console.Write("Масив: ");
for (int i = 0; i < array.Length; i++)
    Console.Write($"{array[i]} ");

//Визначення скільки разів повторюється задане значення в масиві
string value = "int";
int calc = array.ValueCount(value);
```

```
Console.WriteLine($"\nЗначення '{value}' повторюється {calc} разів");

//Видалення повторів масиву
string[] new_n = array.UnoqueArray();
Console.Write("Новий масив: ");
for(int i = 0; i < new_n.Length; i++)
    Console.Write($"{new_n[i]} ");
```

Результат виконання:

Рис.8 – Демонстрація записку програми

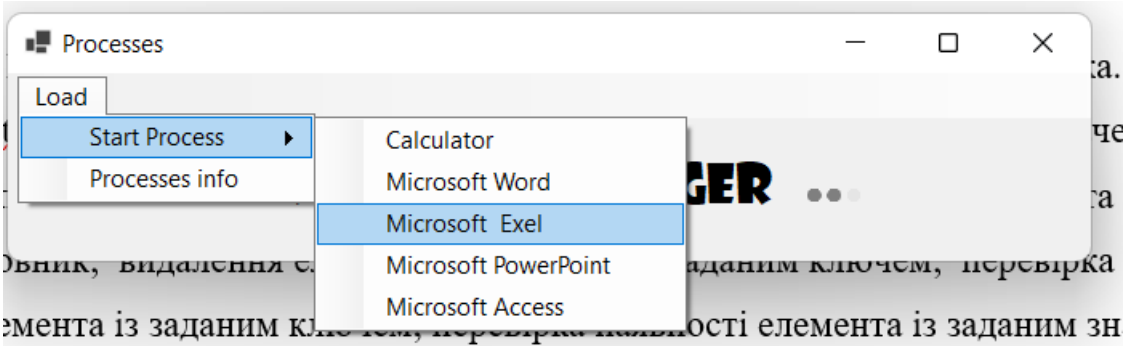
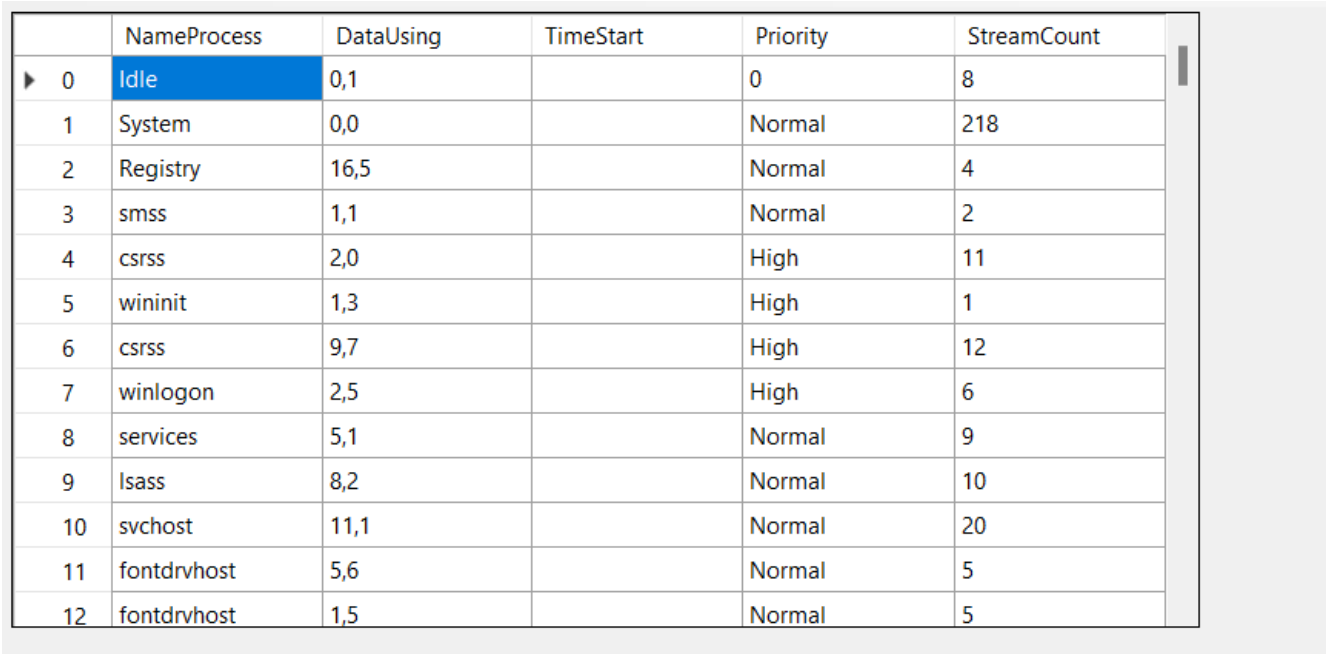


Рис.9 – Демонстрація вибору процесу

Рис.9 – Демонстрація запуску процесу



	NameProcess	DataUsing	TimeStart	Priority	StreamCount
0	Idle	0,1		0	8
1	System	0,0		Normal	218
2	Registry	16,5		Normal	4
3	smss	1,1		Normal	2
4	csrss	2,0		High	11
5	wininit	1,3		High	1
6	csrss	9,7		High	12
7	winlogon	2,5		High	6
8	services	5,1		Normal	9
9	lsass	8,2		Normal	10
10	svchost	11,1		Normal	20
11	fontdrvhost	5,6		Normal	5
12	fontdrvhost	1,5		Normal	5

Рис.10 – Демонстрація всіх процесів

Висновок: На даній лабораторній роботі було засвоєно такі теми, як потоки, клас Thread, процеси та клас Process, клас BackgroundWorker та закріплено основні операції при роботі з ними.