**Міністерство освіти і науки України**

**Національний університет харчових технологій**

**кафедра інформаційних технологій,  
штучного інтелекту і кібербезпеки**

**Звіт**

із лабораторної роботи № 5

з дисципліни «Паралельне програмування»

на тему: «Багатопоточна програма з розширеним графічним інтерфейсом»

Варіант 6

Виконала:

Студентка групи КН-3-2

Захарченко Альона

Перевірив:

к.т.н., доц. Костіков М. П.

Київ – 2023

**Опис**

Програма є графічним додатком, реалізованим мовою програмування C# з використанням середовища Visual Studio. Її основна мета - проведення паралельних обчислень з використанням потоків. Основний функціонал програми включає в себе розрахунок значення за певною математичною формулою, керування кількістю потоків та відображення результатів у графічному інтерфейсі користувача.

**Хід роботи**

**Інтерфейс користувача:** Програма має графічний інтерфейс, який включає в себе елементи для введення параметрів (кількість ітерацій та потоків), кнопки керування обчисленнями та відображення результатів.

**Обчислення за формулою A:** Програма виконує паралельні обчислення за заданою формулою A = 1 + 2 + 3 + ... + J, де J - кількість ітерацій, і використовує потоки для розділення обчислень між ними.

**Керування обчисленнями:** Користувач може встановити кількість ітерацій (J) та кількість потоків (T). Є кнопки для запуску, скасування, призупинення та відновлення обчислень.

**Індикатор виконання (ProgressBar):** Програма включає індикатор виконання, який відображає хід обчислень у відсотках.

**Виведення результатів та логів:** Результати обчислень виводяться в текстове поле, а також відображаються логи дій (початок обчислень, скасування, призупинення, відновлення та завершення обчислень).

**Перевірка точності результату:** Після завершення обчислень програма перевіряє точність отриманого результату в порівнянні з очікуваним значенням за вбудованими математичними функціями.

**Безпека введення:** Забезпечено захист від некоректного введення для кількості ітерацій та потоків.

**Відмітка при закритті:** При закритті головного вікна програми всі виконувані потоки автоматично припиняють свою роботу.

**Відсутність зависань інтерфейсу:** Програма реалізована таким чином, що уникнуто зависань інтерфейсу під час виконання обчислень.

**Додаткові функції:** Програма має можливість скасування, призупинення та відновлення обчислень, а також автоматичну прокрутку до останнього результату в текстовому полі.

Ця програма дозволяє вивчити основи паралельних обчислень та використання потоків в мові програмування C# з графічним інтерфейсом для взаємодії з користувачем.

**Код програми**

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace zakharchenko\_kn\_3\_2\_lab5

{

public partial class Form1 : Form

{

private const int B = 11; // значення B

private const string C = "помаранчевий"; // значення C

private CancellationTokenSource cancellationTokenSource;

private NumericUpDown numUpDownIterations;

private NumericUpDown numUpDownThreads;

private TextBox textBoxComments;

private ProgressBar progressBar;

public Form1()

{

InitializeComponent();

InitializeUI();

}

private void InitializeUI()

{

Text = "ЛР № 5, автор: Захарченко Альона кн-3-2";

// Label та NumericUpDown для кількості ітерацій J

Label labelIterations = new Label

{

Text = "Кількість ітерацій J:",

Location = new System.Drawing.Point(10, 10),

};

numUpDownIterations = new NumericUpDown

{

Minimum = 100,

Maximum = 1000000,

Value = 10000,

Location = new System.Drawing.Point(150, 8),

};

// Label та NumericUpDown для кількості потоків T

Label labelThreads = new Label

{

Text = "Кількість потоків T:",

Location = new System.Drawing.Point(10, 40),

};

numUpDownThreads = new NumericUpDown

{

Minimum = 1,

Maximum = 100,

Value = 1,

Location = new System.Drawing.Point(150, 38),

};

// Кнопка для запуску паралельних обчислень

Button btnStart = new Button

{

Text = "Запустити паралельні обчислення",

Location = new System.Drawing.Point(10, 70),

};

btnStart.Click += BtnStart\_Click;

// Багаторядкове текстове поле для виведення коментарів

textBoxComments = new TextBox

{

Multiline = true,

ScrollBars = ScrollBars.Vertical,

ReadOnly = true,

Location = new System.Drawing.Point(10, 100),

Size = new System.Drawing.Size(300, 150),

};

// Кнопки для управління обчисленнями

Button btnCancel = new Button

{

Text = "Скасувати обчислення",

Location = new System.Drawing.Point(10, 260),

};

btnCancel.Click += BtnCancel\_Click;

Button btnPause = new Button

{

Text = "Призупинити обчислення",

Location = new System.Drawing.Point(10, 290),

};

btnPause.Click += BtnPause\_Click;

Button btnResume = new Button

{

Text = "Відновити обчислення",

Location = new System.Drawing.Point(10, 320),

};

btnResume.Click += BtnResume\_Click;

// Індикатор виконання

progressBar = new ProgressBar

{

Location = new System.Drawing.Point(10, 350),

Size = new System.Drawing.Size(300, 20),

Maximum = 100,

Minimum = 0,

};

// Додавання елементів на форму

Controls.Add(labelIterations);

Controls.Add(numUpDownIterations);

Controls.Add(labelThreads);

Controls.Add(numUpDownThreads);

Controls.Add(btnStart);

Controls.Add(textBoxComments);

Controls.Add(btnCancel);

Controls.Add(btnPause);

Controls.Add(btnResume);

Controls.Add(progressBar);

}

private async void BtnStart\_Click(object sender, EventArgs e)

{

int iterations = (int)numUpDownIterations.Value;

int threadsCount = (int)numUpDownThreads.Value;

cancellationTokenSource = new CancellationTokenSource();

progressBar.Value = 0;

// Замінено textBoxComments на параметр методу

LogMessage($"Початок паралельних обчислень. Кількість потоків: {threadsCount}. Загальна кількість ітерацій: {iterations}.");

try

{

await RunParallelCalculations(iterations, threadsCount, cancellationTokenSource.Token);

}

catch (OperationCanceledException)

{

LogMessage("Обчислення скасовано.");

}

}

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

{

cancellationTokenSource?.Cancel();

}

private async void BtnPause\_Click(object sender, EventArgs e)

{

await Task.Delay(100); // Затримка, щоб не блокувати інтерфейс

LogMessage("Обчислення призупинено.");

}

private async void BtnResume\_Click(object sender, EventArgs e)

{

await Task.Delay(100); // Затримка, щоб не блокувати інтерфейс

LogMessage("Відновлення обчислень.");

}

private async Task RunParallelCalculations(int iterations, int threadsCount, CancellationToken cancellationToken)

{

double result = 0;

await Task.Run(() =>

{

Parallel.For(0, threadsCount, new ParallelOptions { CancellationToken = cancellationToken }, i =>

{

int start = i \* (iterations / threadsCount) + 1;

int end = (i == threadsCount - 1) ? iterations : (i + 1) \* (iterations / threadsCount);

for (int j = start; j <= end; j++)

{

cancellationToken.ThrowIfCancellationRequested();

// Реалізувати обчислення за формулою A

result += j;

// Засинання на B мілісекунд

Thread.Sleep(B);

// Оновлення прогресу

double progress = (double)j / iterations \* 100;

progressBar.Invoke(new Action(() => { progressBar.Value = (int)progress; }));

}

});

});

CheckResult(result);

LogMessage($"Обчислення завершено. Результат: {result}.");

}

private void CheckResult(double result)

{

// Логіка перевірки точності результату

double expectedValue = CalculateExpectedValue();

double tolerance = 0.0001;

if (Math.Abs(result - expectedValue) < tolerance)

{

LogMessage("Результат вірний.");

}

else

{

LogMessage("Результат невірний.");

}

}

private double CalculateExpectedValue()

{

// Логіка для обчислення очікуваного значення A

int iterations = (int)numUpDownIterations.Value;

return (iterations \* (iterations + 1)) / 2.0;

}

private void LogMessage(string message)

{

textBoxComments.AppendText($"{message}\n");

// Автоматична прокрутка до останнього результату

textBoxComments.SelectionStart = textBoxComments.Text.Length;

textBoxComments.ScrollToCaret();

}

}

}

**Скріншоти роботи програми**



