Міністерство освіти і науки України

Дніпровський національний університет ім. Олеся Гончара

Факультет фізики, електроніки та комп’ютерних систем

Кафедра ЕОМ

Звіт

до лабораторної роботи №1

з теми «Потоки»

з дисципліни «Системне програмне»

Виконав:

студент групи КІ-21-2

спеціальності

123 «Комп’ютерна інженерія»

Лебідь Олександр Павлович

Перевірив:

Герасимов Володимир Володимирович

Дніпро

2024

**Мета роботи:** вивчити базову структуру концепції багатопоточності мови C#; навчитися створювати та запускати дочірні потоки.

**Хід роботи**

**Завдання**

1. Виконайте послідовний виклик методу Run() у прикладі 1.1 (без створення потоку). Наочно покажіть різницю між роботою програми з другим потоком і без створення нового потоку.

using System;

namespace l1\_1

{

class Program

{

public static int Run(int a, int b)

{

Thread.Sleep(699);

Console.Write("\t Головний потiк = " + b + "\n" + (b - 20));

a++;

return a;

}

static void Main(string[] args)

{

int mt = 0;

Console.WriteLine("Головний потiк пiдрахунок = " + mt);

Console.WriteLine("Новий потiк запущено.");

Console.Write("Пiдрахунок:");

for (int i = 20; i <= 30; i++)

{

mt = Run(mt, i);

}

Console.WriteLine("\nНовий потiк закiнчено.");

Console.WriteLine("\t Головний потiк пiдрахунок = " + mt);

Console.ReadKey();

}

}

}

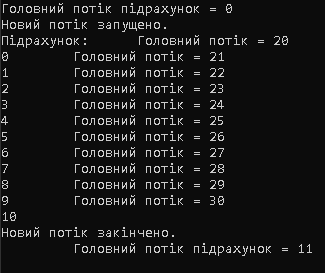


Рис. 1 – Послідовний виклик методу Run() без створення потоку

1. Проаналізуйте та виконайте приклади 1.1-1.4 з практичної частини. Наочно покажіть паралельність виконання потоків.

using System;

using System.Threading;

namespace l1\_3

{

class Program

{

static void Main(string[] args)

{

Thread myThread = new Thread(new ParameterizedThreadStart(Count!));

myThread.Start(); // передаємо змінну у новий потік

for (int i = 1; i <= 10; i++)

{

Console.WriteLine("Потiк 1 \tTime: " + DateTime.Now);

Thread.Sleep(300);

}

Console.ReadLine();

}

public static void Count(object x)

{

for (int i = 1; i <= 10; i++)

{

Console.WriteLine("\tПотiк 2 \t Time:" + DateTime.Now);

Thread.Sleep(400);

}

}

}

}

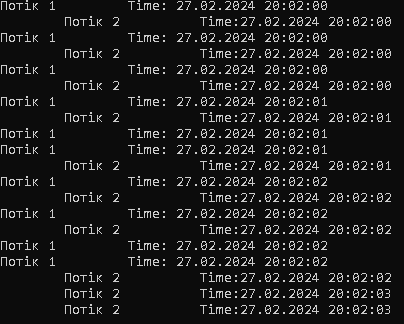


Рис. 2 – Паралельність виконання потоків

1. Виконайте приклади 1.5-1.6 в проектах C# Windows Forms.
2. Виконайте свій варіант відповідно до списку у підгрупи в проекті C# Windows Forms.

**Варіант 5**

Написати програму, в якій перший потік при кожному натисканні на кнопку «Завантажити» читає обраний файл з зображенням і виводить його у вікно програми. При кожному натисканні на кнопку «Поворот», другий потік виконує поворот зображення на 90 градусів.

**Form1.cs:**

using System;

using System.ComponentModel;

using System.Drawing;

using System.Windows.Forms;

using System.IO;

namespace l4\_3.\_6

{

public partial class Form1 : Form

{

private Thread loadThread;

private Thread rotateThread;

private Image image;

public Form1()

{

InitializeComponent();

}

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

{

using (OpenFileDialog dialog = new OpenFileDialog())

{

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

{

loadThread = new Thread(() => loadImage(dialog.FileName));

loadThread.Start();

}

}

}

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

{

if (image != null)

{

rotateThread = new Thread(rotateImage);

rotateThread.Start();

}

}

private void loadImage(string path)

{

image = Image.FromFile(path);

this.Invoke((MethodInvoker)delegate

{

pictureBox1.Image = image;

});

}

private void rotateImage()

{

image.RotateFlip(RotateFlipType.Rotate90FlipNone);

this.Invoke((MethodInvoker)delegate

{

pictureBox1.Image = image;

});

}

}

}

**Form1.cs:**

namespace l4\_3.\_6

{

partial class Form1

{

private System.ComponentModel.IContainer components = null;

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

loadButton = new Button();

rotateButton = new Button();

pictureBox1 = new PictureBox();

((System.ComponentModel.ISupportInitialize)pictureBox1).BeginInit();

SuspendLayout();

loadButton.Location = new Point(544, 94);

loadButton.Name = "loadButton";

loadButton.Size = new Size(75, 23);

loadButton.TabIndex = 0;

loadButton.Text = "Load";

loadButton.UseVisualStyleBackColor = true;

loadButton.Click += loadButton\_Click;

rotateButton.Location = new Point(544, 144);

rotateButton.Name = "rotateButton";

rotateButton.Size = new Size(75, 23);

rotateButton.TabIndex = 1;

rotateButton.Text = "Rotate";

rotateButton.UseVisualStyleBackColor = true;

rotateButton.Click += rotateButton\_Click;

pictureBox1.Location = new Point(81, 37);

pictureBox1.Name = "pictureBox1";

pictureBox1.Size = new Size(421, 363);

pictureBox1.TabIndex = 2;

pictureBox1.TabStop = false;

AutoScaleDimensions = new SizeF(7F, 15F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(800, 450);

Controls.Add(pictureBox1);

Controls.Add(rotateButton);

Controls.Add(loadButton);

Name = "Form1";

Text = "Form1";

((System.ComponentModel.ISupportInitialize)pictureBox1).EndInit();

ResumeLayout(false);

}

private Button loadButton;

private Button rotateButton;

private PictureBox pictureBox1;

}

}

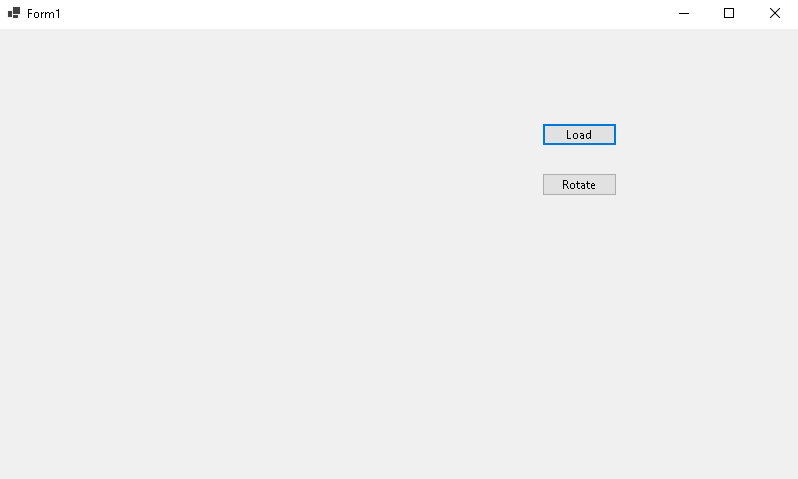


Рис. 3 – Зовнішній вигляд програми

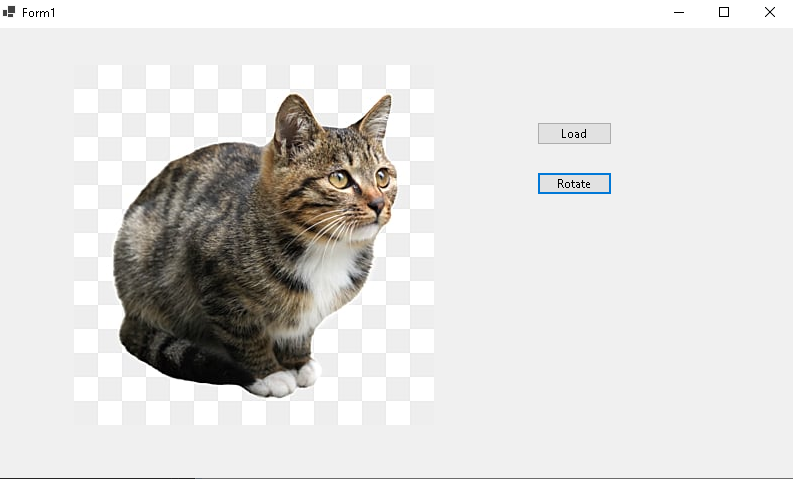


Рис. 4 – Завантаження зображення

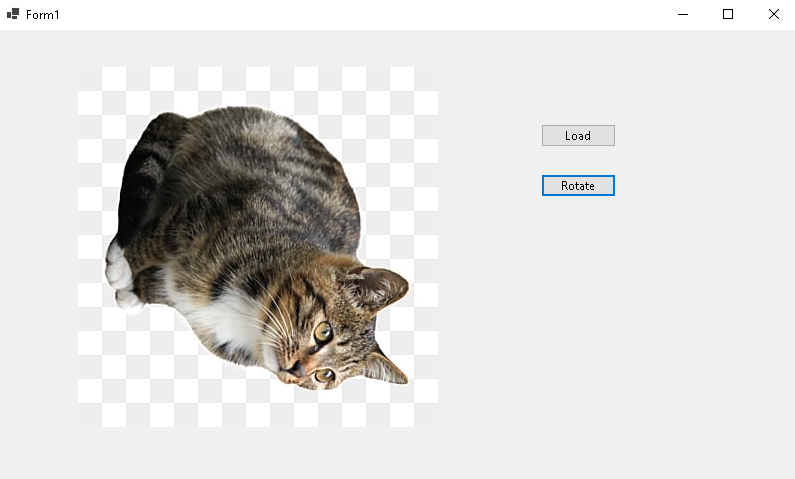


Рис. 5 – Поворот зображення на 90 градусів

**Контрольні питання**

1. **Що таке потік?**

**Потік** – це окремий шлях виконання в програмі. Кожен потік має свій власний стек і не може безпосередньо доступатися до локальних змінних інших потоків.

1. **Чим відрізняється виконання багатопотокової програми від однопотокової?**

Однопотокова програма виконує одну команду за раз. Якщо однопотокова програма виконує складний розрахунок, вся програма зависне, поки розрахунок не буде завершено. Натомість багатопотоковість дозволяє процесу керувати двома або більше одночасними потоками. Кожен потік може обробляти завдання незалежно.

1. **Як створити і запустити потік?**

Щоб створити і запустити потік, треба створити екземпляр класу Thread, передаючи йому делегат ThreadStart, який вказує на метод, що буде виконуватися в новому потоці, а потім викликаєте метод Start().

1. **Як завершити, призупинити і відновити виконання потоку?**

Щоб завершити потік, можна викликати метод Abort() потоку. Однак методи Suspend() і Resume() для призупинення і відновлення потоку вважаються застарілими. Замість цього рекомендується використовувати клас ManualResetEvent.

1. **Як визначити закінчення потоку?**

Щоб визначити закінчення потоку, треба викликати метод Join() потоку.

1. **Як передати аргумент потоку?**

Щоб передати аргумент потоку, треба використовувати делегат ParameterizedThreadStart або передати аргумент методу Start().

1. **Пріоритети потоків. Як поміняти пріоритет потоку?**

Кожен потік має пріоритет, який визначає, як часто потік отримує доступ до ЦП. Можна змінити пріоритет потоку, використовуючи властивість Priority потоку.

1. **Який метод в новому потоці запускає метод Start() класу Thread?**

Метод Start() класу Thread викликає метод run() в новому потоці.

1. **Що робить метод Join() класу Thread?**

Метод Join() класу Thread блокує викликаючий потік, поки потік, метод Join() якого викликано, не завершиться.

**Висновок:** Під час виконання лабораторної роботи було вивчено базову структуру концепції багатопоточності мови C#; вивчено створювання та запускання дочірних потоків.