Міністерство освіти і науки України

Дніпровський національний університет ім. Олеся Гончара

Факультет фізики, електроніки та комп’ютерних систем

Кафедра ЕОМ

Звіт

до лабораторної роботи №3

з теми «Синхронізація»

з дисципліни «Системне програмне забезпечення»

Виконав:

студент групи КІ-21-2

спеціальності

123 «Комп’ютерна інженерія»

Лебідь Олександр Павлович

Перевірив:

Герасимов Володимир Володимирович

Дніпро

2024

**Мета роботи:** ознайомитися з об’єктами синхронізації та навчитися використовувати їх.

**Хід роботи**

**Завдання**

1. Виконати усі приклади розглянутих програм.

2. Виконати усі приклади програм для 2, 3 и N процесів і/або потоків.

Наглядно показати паралельність виконання і синхронізацію процесів і/або потоків.

3. Виконати усі приклади програм для 2, 3 и N процесів і/або потоків в проектах C# Windows Forms.

4. Виконати свій варіант по списку підгрупи в проекті C# Windows Forms:

Варіант 5

Написати програму, яка створює 2 потоки. Перший потік читає поточні

координати миші і зберігає їх у пам'ять. Другий потік малює на екрані коло і читає значення координат миші з пам'яті. При наближенні курсора миші до кола, пересунути коло на деяку відстань у протилежний бік.

**Form1.cs:**  
namespace n3l4

{

public partial class Form1 : Form

{

private Thread mouseThread;

private Thread circleThread;

private Point mousePosition;

private Point circlePosition;

private const int circleRadius = 200;

public Form1()

{

InitializeComponent();

mouseThread = new Thread(TrackMouse);

circleThread = new Thread(DrawCircle);

mouseThread.Start();

circleThread.Start();

}

private void TrackMouse()

{

while (true)

{

mousePosition = this.PointToClient(Cursor.Position);

Thread.Sleep(10);

}

}

private void DrawCircle()

{

while (true)

{

this.Invalidate();

Thread.Sleep(10);

}

}

protected override void OnPaint(PaintEventArgs e)

{

base.OnPaint(e);

if (circlePosition == null)

{

circlePosition = new Point(this.Width / 2, this.Height / 2);

}

if (Math.Sqrt(Math.Pow(mousePosition.X - circlePosition.X, 2) + Math.Pow(mousePosition.Y - circlePosition.Y, 2)) < circleRadius)

{

// Move the circle away from the mouse

circlePosition.X += (circlePosition.X - mousePosition.X) / 2;

circlePosition.Y += (circlePosition.Y - mousePosition.Y) / 2;

}

e.Graphics.FillEllipse(Brushes.Red, circlePosition.X - circleRadius, circlePosition.Y - circleRadius, circleRadius \* 2, circleRadius \* 2);

}

}

}

**Form1.Designer.cs:**

namespace n3l4

{

partial class Form1

{

/// <summary>

/// Required designer variable.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Clean up any resources being used.

/// </summary>

/// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>

protected override void Dispose(bool disposing)

{

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

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Windows Form Designer generated code

/// <summary>

/// Required method for Designer support - do not modify

/// the contents of this method with the code editor.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.ClientSize = new System.Drawing.Size(800, 450);

this.Text = "Form1";

}

#endregion

}

}

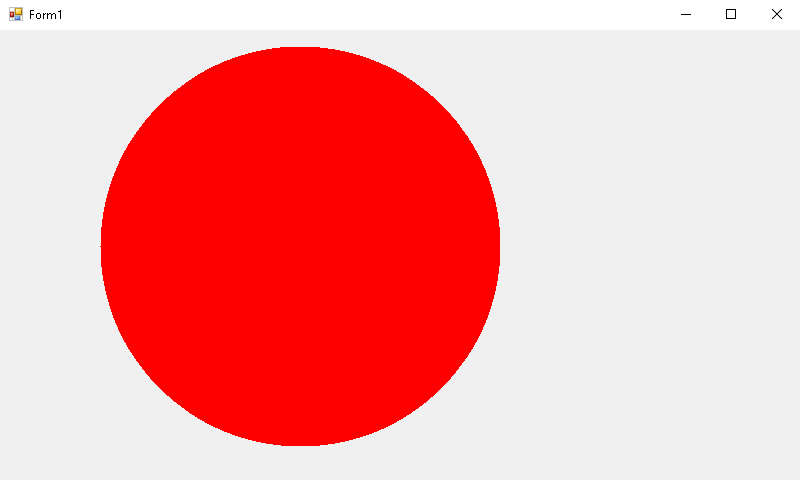


Рис. 1 – Результат програми

**Контрольні питання**

1. **Що таке синхронізація?**

**Синхронізація** – це координація виконання потоків, щоб запобігти конфліктам при доступі до спільних ресурсів або для виконання операцій у визначеному порядку.

1. **Навіщо і коли потрібна синхронізація? Наведіть приклади.**

Синхронізація потрібна, коли два або більше потоки використовують спільний ресурс. Наприклад, якщо два потоки читають та записують в одну і ту ж змінну, можуть виникнути проблеми з непередбачуваною поведінкою.

1. **Які існують об’єкти синхронізації?**

Об’єкти синхронізації включають м’ютекси, семафори, події та монітори.

1. **Що таке м’ютекс?**

**М’ютекс** – це об’єкт синхронізації, який дозволяє кільком потокам контролювати доступ до спільного ресурсу, але в один момент часу лише один потік може володіти м’ютексом.

1. **Що таке семафор?**

**Семафор** – це об’єкт синхронізації, який контролює доступ до пулу ресурсів. Він відрізняється від м’ютекса тим, що дозволяє більше одного потоку мати доступ до ресурсу в один момент часу.

1. **Що таке події?**

**Події** – це об’єкти синхронізації, які дозволяють одному потоку повідомляти іншому про те, що певна подія відбулася.

1. **Методи класу Monitor, що використовуються для синхронізації.**

Клас Monitor використовується для синхронізації потоків. Деякі методи цього класу включають Enter, Exit, Wait, Pulse та PulseAll.

1. **Чим семафор відрізняється від події?**

Головна відмінність між семафором та подією полягає в тому, що семафор дозволяє більше одного потоку мати доступ до ресурсу в один момент часу, тоді як подія дозволяє лише одному потоку мати доступ до ресурсу. Подія також може бути встановлена вручну або автоматично, тоді як семафор завжди встановлюється автоматично.

**Висновок:** Під час виконання лабораторної роботи було ознайомлено з об’єктами синхронізації та вивчено їх використовування.