**ЛАБОРАТОРНА РОБОТА № 3**

з дисципліни

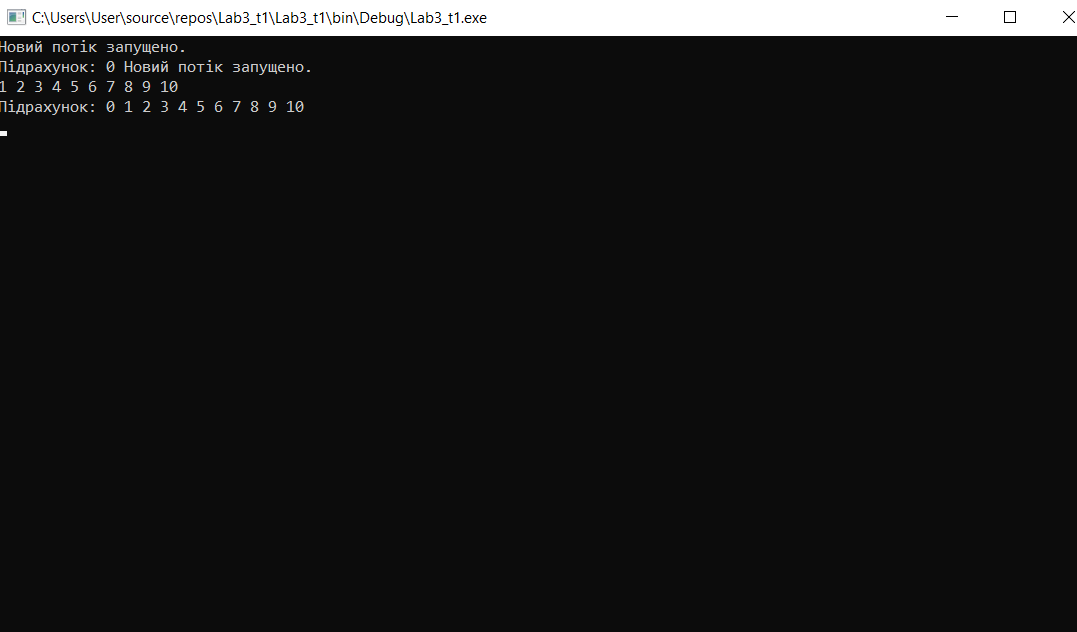
«Системне програмне забезпечення»

Шейко Р. О.

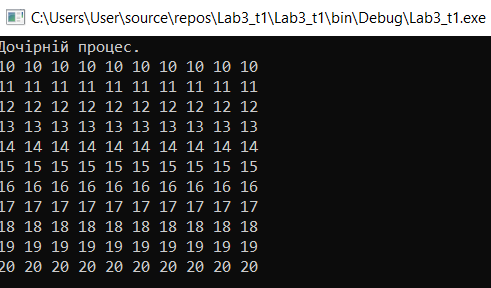
Групи КІ-21-2

1. Виконати усі приклади розглянутих програм.

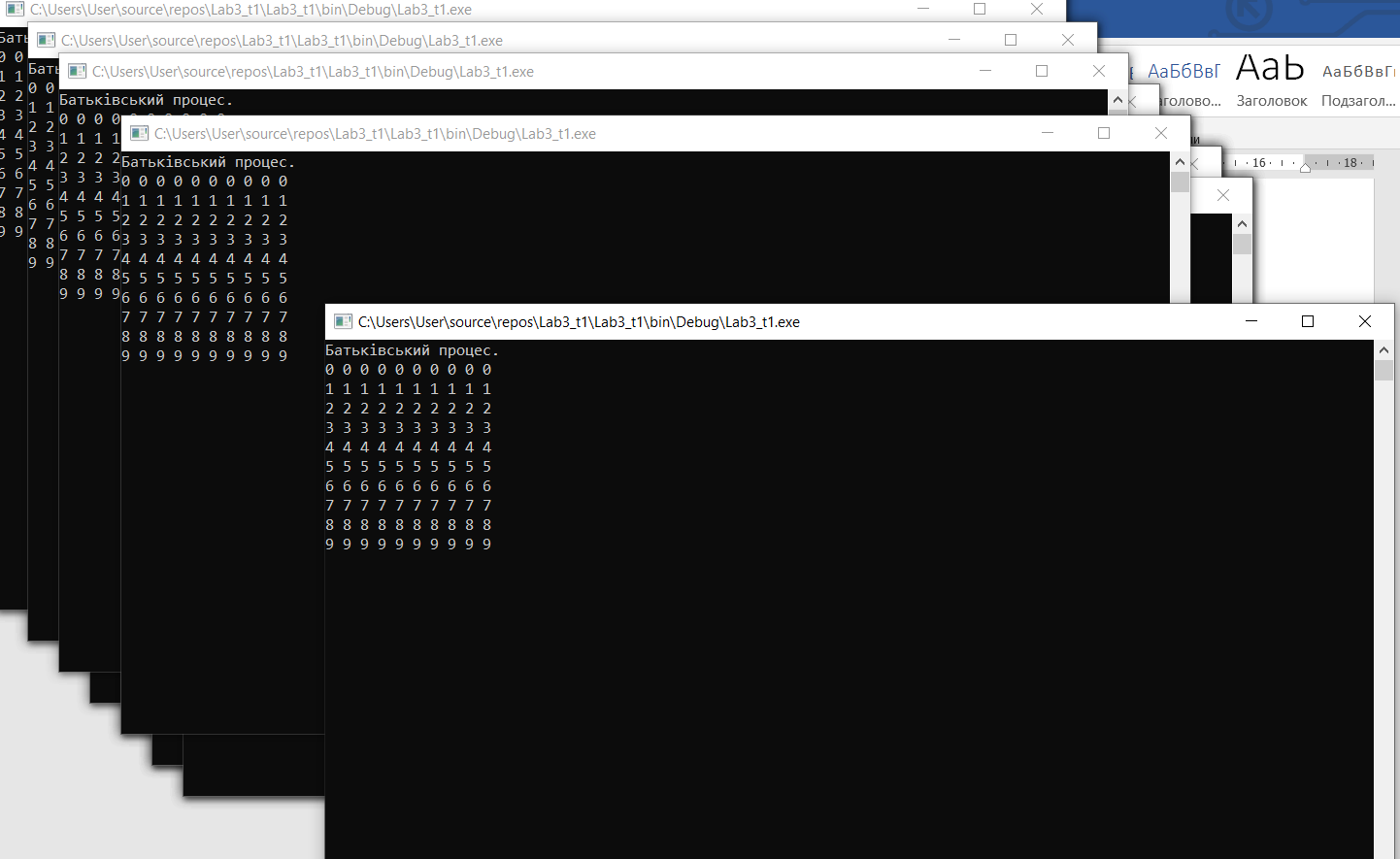
Запустимо лістинг 3.1 з посібника



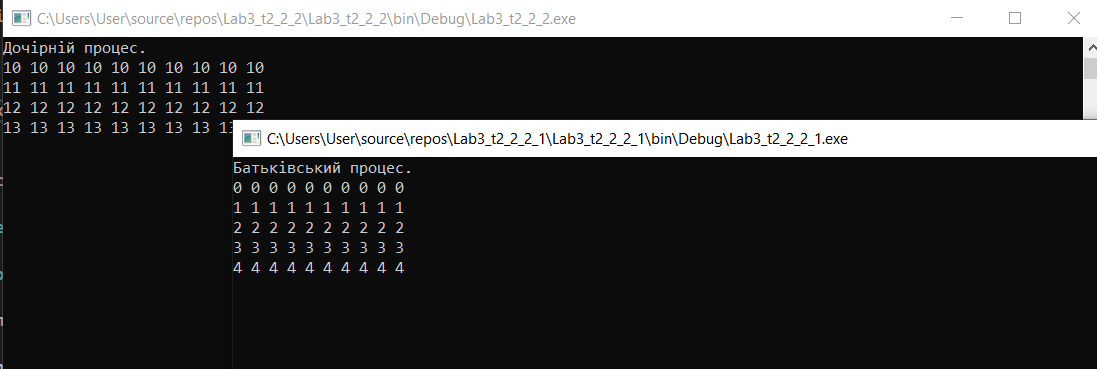
Запустимо лістинг 3.2 з посібника



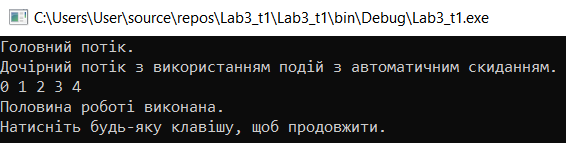
Запустимо лістинг 3.2.1 з посібника



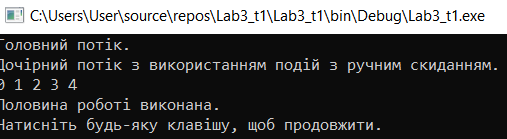
Запустимо лістинги 3.2.1.1 і 3.2.1.2

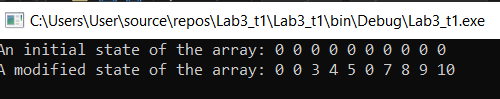


Запустимо лістинг 3.3 з посібника

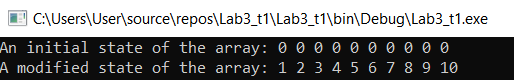


Запустимо лістинг 3.4 з посібника



Запустимо лістинг 3.5 з посібника 

Запустимо лістинг 3.5.1 з посібника



1. Виконати усі приклади програм для 2, 3 и N процесів і/або потоків. Наглядно показати паралельність виконання і синхронізацію процесів і/або потоків.

*Лістинг 2.1*

using System;

using System.Threading;

namespace l\_3\_1

{

class Program

{

static object lockOn = new object();

static public void Run()

{

Thread.Sleep(200);

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

lock (lockOn)

{

Console.Write("Пiдрахунок: ");

for (int i = 0; i <= 10; i++)

{

Console.Write(i + " ");

Thread.Sleep(10);

}

Console.WriteLine();

}

}

static void Main(string[] args)

{

Console.Write("Введiть кiлькiсть потокiв: ");

int n = int.Parse(Console.ReadLine());

Thread[] threads = new Thread[n];

for (int i = 0; i < n; i++)

{

threads[i] = new Thread(Run);

threads[i].Start();

}

for (int i = 0; i < n; i++)

{

threads[i].Join();

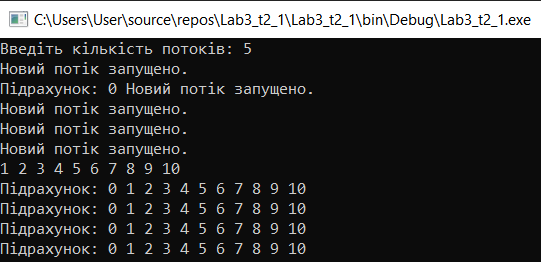
}

Console.ReadKey();

}

}

}



*Лістинг 2.2*

using System;

using System.Threading;

namespace l\_3\_2

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите количество потоков:");

int N = int.Parse(Console.ReadLine());

Thread[] threads = new Thread[N];

for (int k = 0; k < N; k++)

{

threads[k] = new Thread(() =>

{

Console.WriteLine("Дочiрнiй процес.");

for (int j = 10; j <= 20; j++)

{

for (int i = 0; i < 10; i++)

{

Console.Write(j + " ");

Thread.Sleep(100);

}

Console.WriteLine();

}

});

threads[k].Start();

}

foreach (Thread t in threads)

{

t.Join();

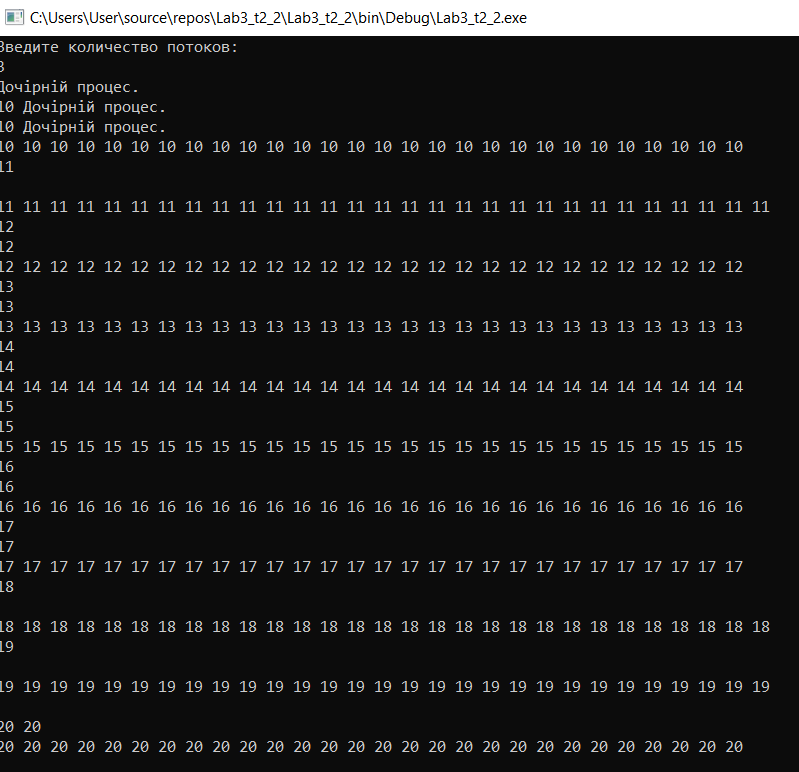
}

Console.ReadKey();

}

}

}



*Лістинг 2.2.1*

using System;

using System.Diagnostics;

using System.Threading;

namespace l\_3\_2\_p

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите количество потоков:");

int N = int.Parse(Console.ReadLine());

Console.WriteLine("Батькiвський процес.");

string appName =

@"C:\Users\User\source\repos\Lab3\_t2\_2\Lab3\_t2\_2\bin\Debug\Lab3\_t2\_2.exe";

Process[] newProcs = new Process[N];

for (int k = 0; k < N; k++)

{

newProcs[k] = new Process();

newProcs[k].StartInfo.FileName = appName;

newProcs[k].Start();

}

for (int j = 0; j < 10; ++j)

{

for (int i = 0; i < 10; i++)

{

Console.Write(j + " ");

Thread.Sleep(100);

}

Console.WriteLine();

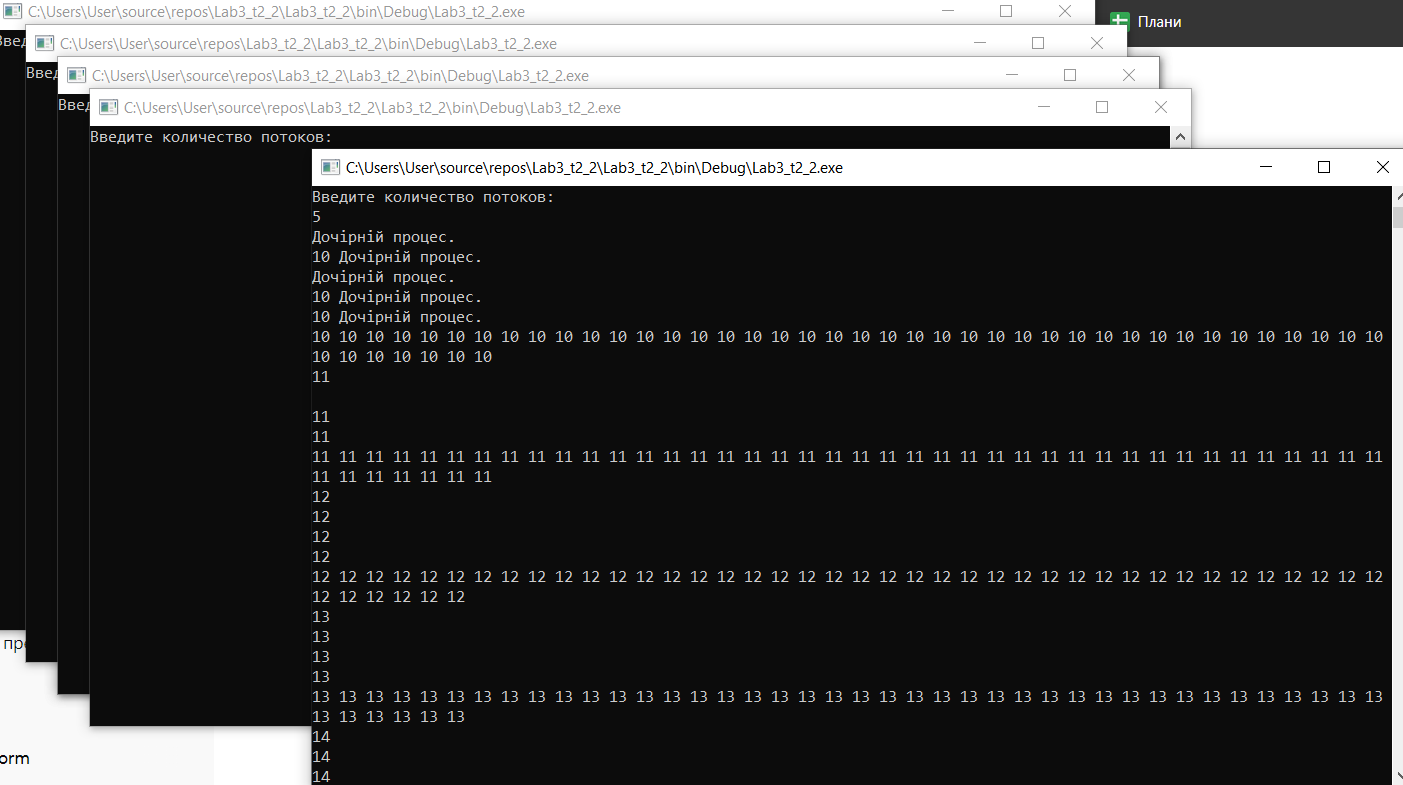
}

Console.ReadKey();

}

}

}



*Лістинг 2.2.2*

using System;

namespace Lab3\_t2\_2\_2

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Дочiрнiй процес.");

if (args.Length < 1)

{

Console.WriteLine("No arguments.");

Console.ReadKey();

return;

}

if (int.TryParse(args[0], out int N))

{

for (int j = 0; j < N; j++)

{

for (int i = 0; i < 10; i++)

{

Console.Write(j + " ");

System.Threading.Thread.Sleep(100);

}

Console.WriteLine();

System.Threading.Thread.Sleep(10);

}

}

Console.ReadKey();

}

}

}

*Лістинг 2.2.2.1*

using System;

using System.Diagnostics;

using System.IO;

using System.IO.Pipes;

namespace l\_3\_2\_1\_p

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Батькiвський процес.");

string mtxName = "MyMutex";

string appName = @"C:\Users\User\source\repos\Lab3\_t2\_2\_2\Lab3\_t2\_2\_2\bin\Debug\Lab3\_t2\_2\_2.exe";

using (var pipeServer = new AnonymousPipeServerStream(PipeDirection.Out, HandleInheritability.Inheritable))

{

using (var sw = new StreamWriter(pipeServer))

{

var psi = new ProcessStartInfo(appName, "10")

{

RedirectStandardOutput = true,

UseShellExecute = false

};

Process p = Process.Start(psi);

string line;

while ((line = p.StandardOutput.ReadLine()) != null)

{

Console.WriteLine(line);

}

}

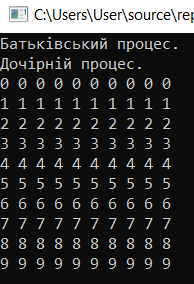
}

Console.ReadKey();

}

}

}



*Лістинг 2.3*

using System;

using System.Threading;

namespace l\_3\_3

{

class Program

{

static AutoResetEvent newEvnt = new AutoResetEvent(false);

static void Run()

{

Console.WriteLine("Дочiрний потiк з використанням подiй з автоматичним скиданням.");

for (int i = 0; i <= 10; i++)

{

Console.Write(i + " ");

if (i == 4)

{

newEvnt.Set();

Thread.Sleep(100);

newEvnt.WaitOne();

Console.WriteLine();

}

}

}

static void Main(string[] args)

{

Console.WriteLine("Головний потiк.");

Console.Write("Введiть кiлькiсть потокiв: ");

int N = int.Parse(Console.ReadLine());

Thread[] threads = new Thread[N];

for (int i = 0; i < N; i++)

{

threads[i] = new Thread(Run);

threads[i].Start();

newEvnt.WaitOne();

Console.WriteLine("\nПоловина роботi виконана.");

Console.WriteLine("Натиснiть будь-яку клавiшу, щоб продовжити.");

Console.ReadKey();

newEvnt.Set();

}

for (int i = 0; i < N; i++)

{

threads[i].Join();

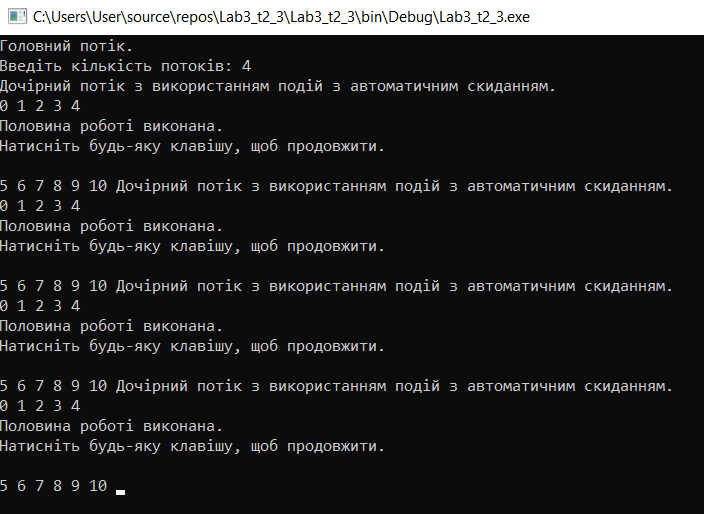
}

Console.ReadKey();

}

}

}



*Лістинг 2.4*

using System;

using System.Threading;

namespace l\_3\_4

{

class Program

{

static ManualResetEvent[] events;

static int N;

static void Run(object index)

{

int i = 0;

int threadIndex = (int)index;

Console.WriteLine($"Дочірній потік {threadIndex} з використанням подій з ручним скиданням.");

for (i = 0; i <= 10; i++)

{

Console.Write(i + " ");

if (i == 4)

{

events[threadIndex].Set();

Thread.Sleep(100);

events[threadIndex].WaitOne();

Console.WriteLine();

}

}

if (i == 11)

{

events[threadIndex].Set();

}

}

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.UTF8; // Установка кодировки UTF-8

Console.WriteLine("Введiть кiлькiсть потокiв (N):");

N = int.Parse(Console.ReadLine());

events = new ManualResetEvent[N];

Console.WriteLine("Головний потік.");

Thread[] threads = new Thread[N];

for (int i = 0; i < N; i++)

{

events[i] = new ManualResetEvent(false);

threads[i] = new Thread(Run);

threads[i].Start(i);

}

for (int i = 0; i < N; i++)

{

events[i].WaitOne();

events[i].Reset();

}

Console.WriteLine("\nПоловина роботи виконана.");

Console.WriteLine("Натисніть будь-яку клавішу, щоб продовжити.");

Console.ReadKey();

for (int i = 0; i < N; i++)

{

events[i].Set();

threads[i].Join();

}

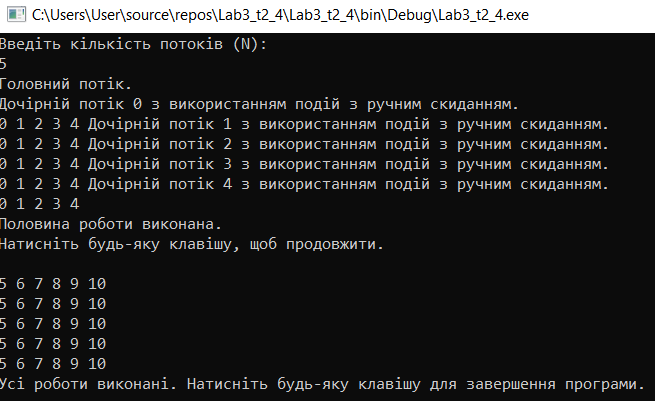
Console.WriteLine("\nУсі роботи виконані. Натисніть будь-яку клавішу для завершення програми.");

Console.ReadKey();

}

}

}



*Лістинг 2.5*

using System;

using System.Threading;

namespace l\_3\_5

{

class Program

{

static int[] arr = new int[10];

static int N;

static void Run()

{

for (int i = 0; i < 10; i++)

{

arr[i] = i + 1;

Thread.Sleep(70);

}

}

static void ModifyArray(object id)

{

int threadId = (int)id;

for (int i = 0; i < 10; i++)

{

arr[i] += threadId;

Thread.Sleep(70);

}

}

static void Main(string[] args)

{

Console.OutputEncoding = System.Text.Encoding.UTF8;

Console.Write("Введіть кількість потоків: ");

N = int.Parse(Console.ReadLine());

Console.Write("Початковий стан масиву: ");

for (int i = 0; i < 10; i++)

Console.Write(arr[i] + " ");

Console.WriteLine();

Thread[] threads = new Thread[N];

for (int i = 0; i < N; i++)

{

threads[i] = new Thread(ModifyArray);

threads[i].Start(i + 1);

}

for (int i = 0; i < N; i++)

threads[i].Join();

Console.Write("Модифікований стан масиву: ");

for (int i = 0; i < 10; i++)

{

Console.Write(arr[i] + " ");

Thread.Sleep(70);

}

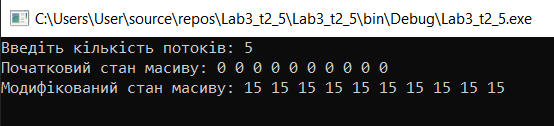
Console.WriteLine();

Console.ReadKey();

}

}

}



*Лістинг 2.5.1*

using System;

using System.Threading;

namespace l\_3\_5\_1

{

class Program

{

static Semaphore[] semaphores;

static int[] arr = new int[10];

static void Run(object semaphore)

{

Semaphore sem = (Semaphore)semaphore;

for (int i = 0; i < 10; i++)

{

arr[i] = i + 1;

sem.Release();

Thread.Sleep(500);

}

}

static void Main(string[] args)

{

Console.Write("Enter the number of threads: ");

int N = Convert.ToInt32(Console.ReadLine());

semaphores = new Semaphore[N];

for (int i = 0; i < N; i++)

{

semaphores[i] = new Semaphore(0, 1);

}

Console.Write("An initial state of the array: ");

for (int i = 0; i < 10; i++)

{

Console.Write(arr[i] + " ");

}

Console.WriteLine();

Thread[] threads = new Thread[N];

for (int i = 0; i < N; i++)

{

threads[i] = new Thread(Run);

threads[i].Start(semaphores[i]);

}

Console.Write("A modified state of the array: ");

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < N; j++)

{

semaphores[j].WaitOne();

Console.Write(arr[i] + " ");

}

}

Console.WriteLine();

for (int i = 0; i < N; i++)

{

threads[i].Join();

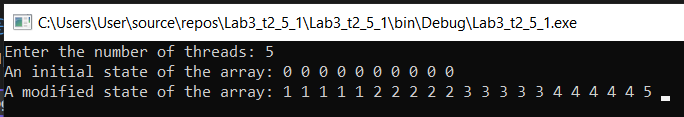
}

Console.ReadKey();

}

}

}



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

*Лістинг 3.1*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Lab3\_t3\_1

{

public partial class Form1 : Form

{

private static readonly object lockOn = new object();

public Form1()

{

InitializeComponent();

}

private void AppendText(string text)

{

if (InvokeRequired)

{

BeginInvoke(new Action<string>(AppendText), text);

return;

}

textBox1.AppendText(text + Environment.NewLine);

}

private void Run()

{

Thread.Sleep(200);

AppendText("Новий потік запущено.");

lock (lockOn)

{

AppendText("Підрахунок: ");

for (int i = 0; i <= 10; i++)

{

AppendText(i + " ");

Thread.Sleep(10);

}

AppendText("");

}

}

private void button1\_Click(object sender, EventArgs e)

{

int n;

if (int.TryParse(textBox2.Text, out n))

{

Thread[] threads = new Thread[n];

for (int i = 0; i < n; i++)

{

threads[i] = new Thread(Run);

threads[i].Start();

}

for (int i = 0; i < n; i++)

{

threads[i].Join();

}

}

else

{

MessageBox.Show("Введено некоректне значення для кількості потоків.", "Помилка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

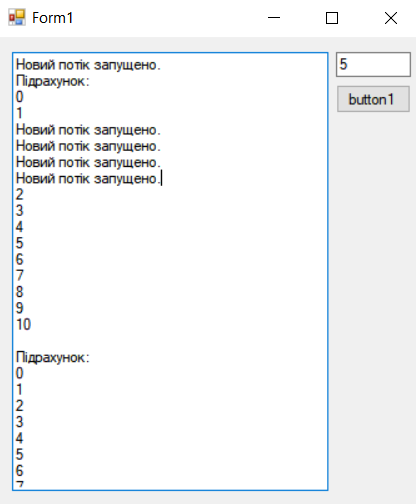
private void Form1\_Load(object sender, EventArgs e)

{

}

}

}



*Лістинг 3.2*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Lab3\_t3\_2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private async void button1\_Click(object sender, EventArgs e)

{

int N = Convert.ToInt32(textBox1.Text);

await Task.Run(() => StartThreads(N));

}

private void StartThreads(int N)

{

Thread[] threads = new Thread[N];

for (int k = 0; k < N; k++)

{

threads[k] = new Thread(() =>

{

for (int j = 10; j <= 20; j++)

{

for (int i = 0; i < 10; i++)

{

AddTextToOutput(j + " ");

Task.Delay(100).Wait();

}

AddTextToOutput(Environment.NewLine);

}

});

threads[k].Start();

}

foreach (Thread t in threads)

{

t.Join();

}

}

private void AddTextToOutput(string text)

{

if (textBox2.InvokeRequired)

{

textBox2.Invoke(new Action<string>(AddTextToOutput), text);

}

else

{

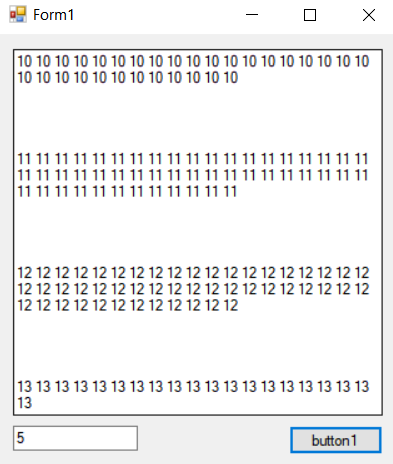
textBox2.AppendText(text);

}

}

}

}



*Лістинг 3.2.1*

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 static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Lab3\_t3\_2\_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

int N = int.Parse(textBox1.Text);

MessageBox.Show("Батькiвський процес.");

for (int k = 0; k < N; k++)

{

Process.Start(@"C:\Users\User\source\repos\Lab3\_t2\_2\Lab3\_t2\_2\bin\Debug\Lab3\_t2\_2.exe");

}

string outputText = "Батькiвський процес.";

for (int j = 0; j < 10; ++j)

{

for (int i = 0; i < 10; i++)

{

outputText += j + " ";

System.Threading.Thread.Sleep(100);

}

outputText += "\n";

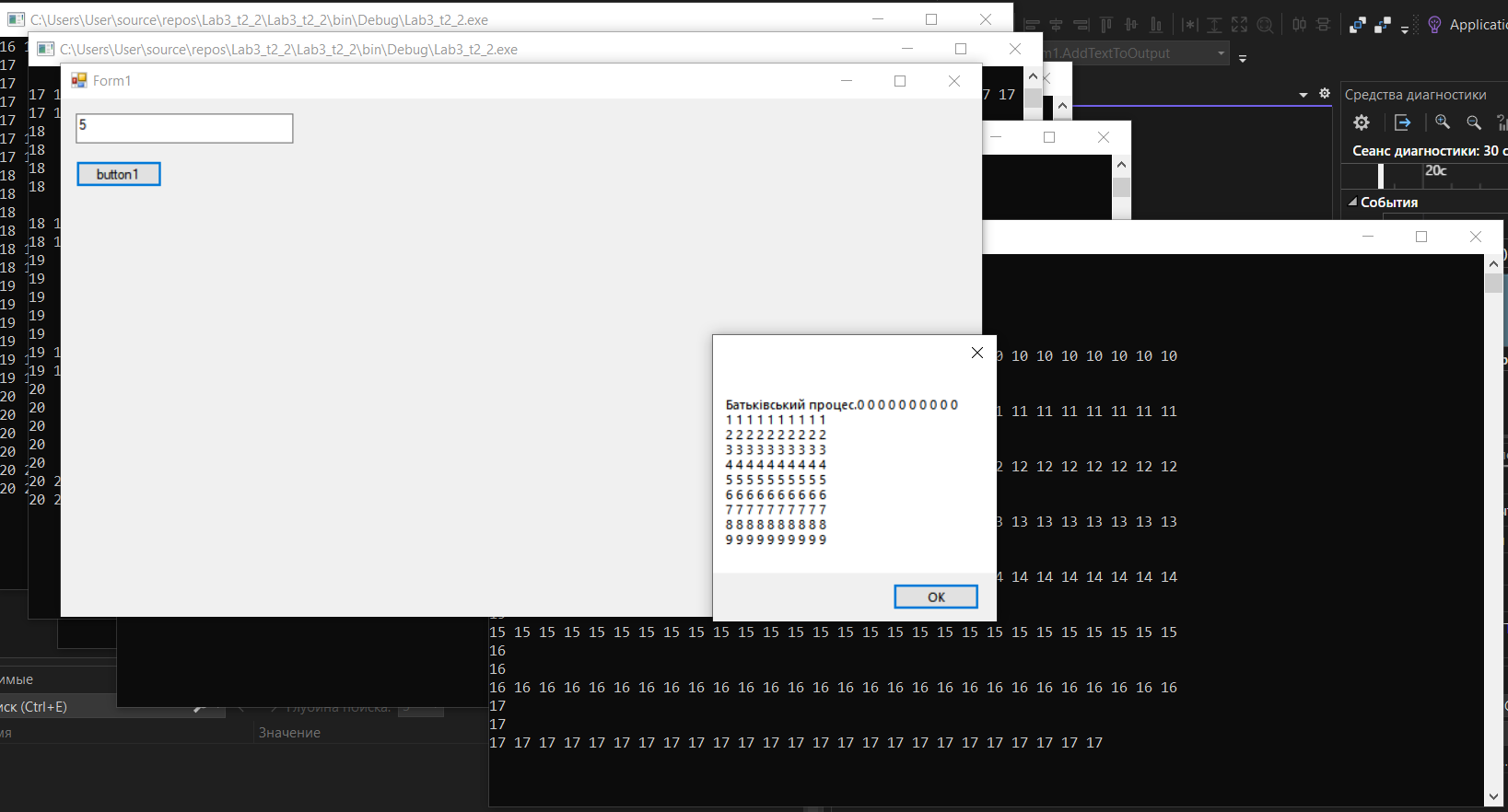
}

MessageBox.Show(outputText);

}

}

}



*Лістинг 3.2.2*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Lab3\_t3\_2\_2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

string[] args = Environment.GetCommandLineArgs();

if (args.Length < 2)

{

MessageBox.Show("No arguments.");

return;

}

if (int.TryParse(args[1], out int N))

{

for (int j = 0; j < N; j++)

{

for (int i = 0; i < 10; i++)

{

UpdateText(j.ToString() + " ");

Thread.Sleep(100);

}

UpdateText(Environment.NewLine);

Thread.Sleep(10);

}

}

}

private void UpdateText(string text)

{

if (textBox1.InvokeRequired)

{

textBox1.Invoke(new Action(() => { textBox1.Text += text; }));

}

else

{

textBox1.Text += text;

}

}

}

}

*Лістинг 3.2.2.1*

using System;

using System.Diagnostics;

using System.IO;

using System.IO.Pipes;

using System.Threading;

using System.Windows.Forms;

namespace Lab3\_t3\_2\_2\_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

int N;

if (!int.TryParse(textBox1.Text, out N))

{

MessageBox.Show("Invalid number of threads.");

return;

}

string childAppPath = @"C:\Users\User\source\repos\Lab3\_t2\_2\_2\Lab3\_t2\_2\_2\bin\Debug\Lab3\_t2\_2\_2.exe";

using (var pipeServer = new AnonymousPipeServerStream(PipeDirection.Out, HandleInheritability.Inheritable))

{

var psi = new ProcessStartInfo(childAppPath, pipeServer.GetClientHandleAsString())

{

UseShellExecute = false

};

Process p = Process.Start(psi);

using (var sw = new StreamWriter(pipeServer))

{

}

p.WaitForExit();

}

using (var pipeServer = new AnonymousPipeServerStream(PipeDirection.Out, HandleInheritability.Inheritable))

{

using (var sr = new StreamReader(pipeServer))

{

var psi = new ProcessStartInfo(childAppPath, N.ToString())

{

RedirectStandardOutput = true,

UseShellExecute = false

};

Process p = Process.Start(psi);

ThreadPool.QueueUserWorkItem((\_) =>

{

string line;

while ((line = sr.ReadLine()) != null)

{

UpdateText(line);

}

});

p.WaitForExit();

}

}

}

private void UpdateText(string text)

{

if (textBox2.InvokeRequired)

{

textBox2.Invoke(new Action(() => { textBox2.AppendText(text + Environment.NewLine); }));

}

else

{

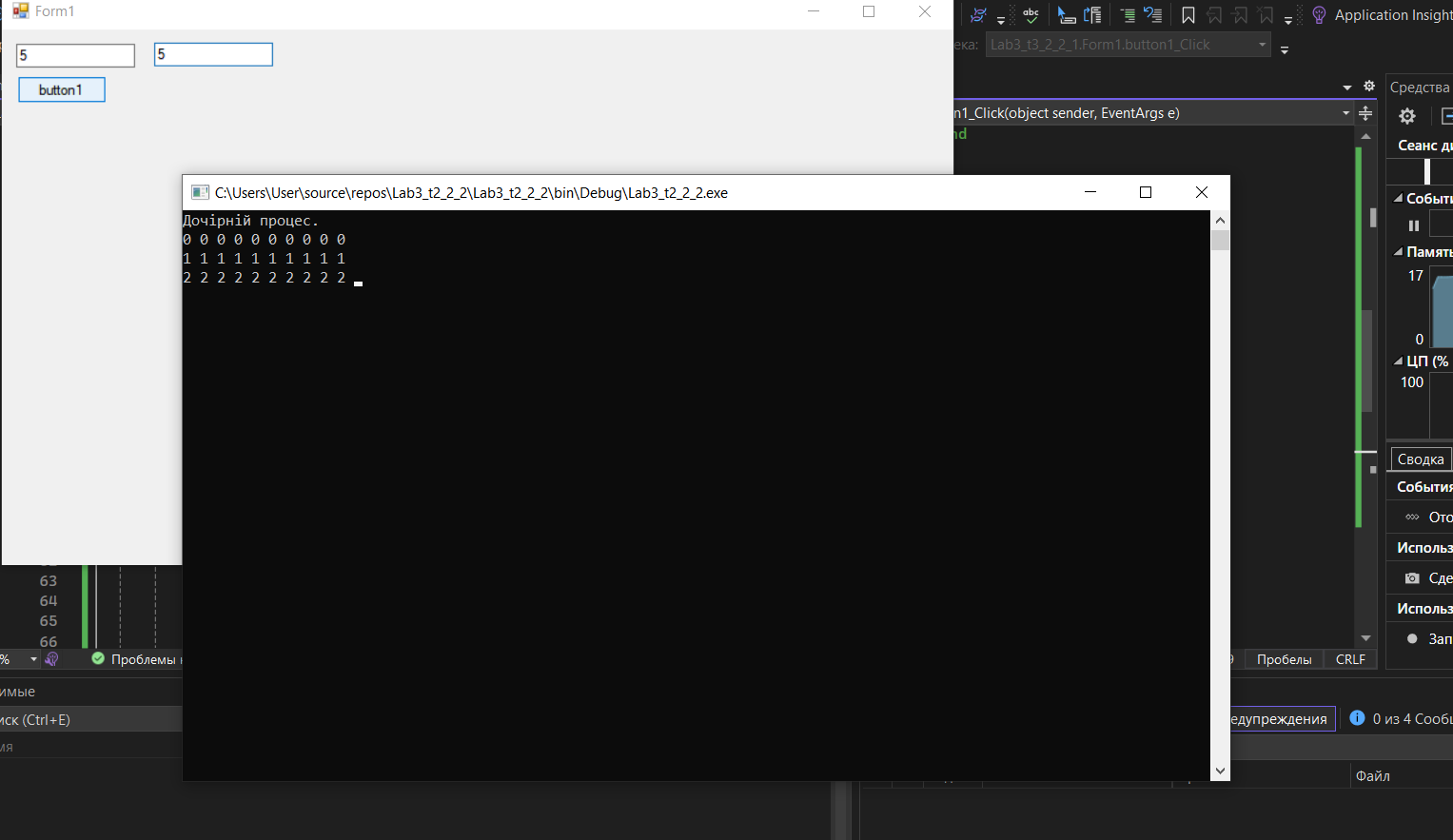
textBox2.AppendText(text + Environment.NewLine);

}

}

}

}



*Лістинг 3.3*

using System;

using System.ComponentModel;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab3\_t3\_3

{

public partial class Form1 : Form

{

private AutoResetEvent newEvnt = new AutoResetEvent(false);

private BackgroundWorker worker;

private int currentThreadIndex = 0;

public Form1()

{

InitializeComponent();

InitializeBackgroundWorker();

}

private void InitializeBackgroundWorker()

{

worker = new BackgroundWorker();

worker.DoWork += Worker\_DoWork;

worker.RunWorkerCompleted += Worker\_RunWorkerCompleted;

}

private async void Worker\_DoWork(object sender, DoWorkEventArgs e)

{

int N = (int)e.Argument;

for (int i = 0; i < N; i++)

{

for (int j = 0; j <= 10; j++)

{

await AppendTextAsync(textBox2, $"{j} ");

if (j == 4)

{

newEvnt.Set();

await Task.Delay(100);

newEvnt.WaitOne();

await AppendTextAsync(textBox2, Environment.NewLine);

}

}

if (i == N - 1)

{

button1.Invoke(new Action(() =>

{

button1.Enabled = false;

}));

}

else

{

await AppendTextAsync(textBox2, "\nПоловина роботi виконана.\n");

await AppendTextAsync(textBox2, "Натиснiть кнопку, щоб продовжити.\n");

i++;

newEvnt.WaitOne();

}

}

}

private async Task AppendTextAsync(TextBox textBox, string text)

{

if (textBox.InvokeRequired)

{

await Task.Run(() =>

{

textBox.BeginInvoke(new Action(() =>

{

textBox.AppendText(text);

}));

});

}

else

{

textBox.AppendText(text);

}

}

private void Worker\_RunWorkerCompleted(object sender, RunWorkerCompletedEventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

int N;

if (!int.TryParse(textBox1.Text, out N))

{

MessageBox.Show("Invalid number of threads.");

return;

}

if (!worker.IsBusy)

{

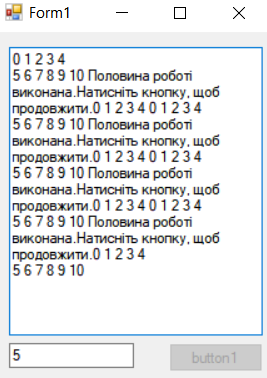
worker.RunWorkerAsync(N);

}

}

}

}



*Лістинг 3.4*

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab3\_t3\_4

{

public partial class Form1 : Form

{

private ManualResetEvent[] events;

private int N;

public Form1()

{

InitializeComponent();

}

private void Run(object index)

{

int threadIndex = (int)index;

int i = 0;

AppendText($"Дочірній потік {threadIndex} з використанням подій з ручним скиданням.");

for (i = 0; i <= 10; i++)

{

AppendText(i + " ");

if (i == 4)

{

events[threadIndex].Set();

Thread.Sleep(100);

events[threadIndex].WaitOne();

AppendText(Environment.NewLine);

}

}

if (i == 11)

{

events[threadIndex].Set();

}

}

private void AppendText(string text)

{

if (textBox2.InvokeRequired)

{

textBox2.BeginInvoke(new Action(() =>

{

textBox2.AppendText(text);

}));

}

else

{

textBox2.AppendText(text);

}

}

private async void button1\_Click(object sender, EventArgs e)

{

if (!int.TryParse(textBox1.Text, out N))

{

MessageBox.Show("Invalid number of threads.");

return;

}

events = new ManualResetEvent[N];

Task[] tasks = new Task[N];

for (int i = 0; i < N; i++)

{

events[i] = new ManualResetEvent(false);

int indexCopy = i; // Создаем копию переменной для передачи в метод Run

tasks[i] = Task.Run(() => Run(indexCopy));

}

await Task.WhenAll(tasks);

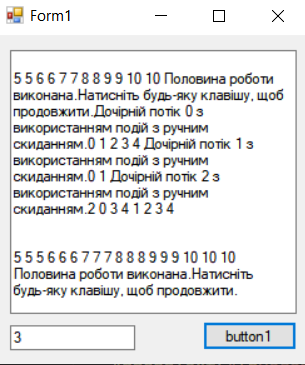
AppendText("\nПоловина роботи виконана.\n");

AppendText("Натисніть будь-яку клавішу, щоб продовжити.\n");

}

}

}



*Лістинг 3.5*

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Lab3\_t5

{

public partial class Form1 : Form

{

private int[] arr = new int[10];

private int N;

public Form1()

{

InitializeComponent();

}

private void Run()

{

for (int i = 0; i < 10; i++)

{

arr[i] = i + 1;

Thread.Sleep(70);

}

}

private void ModifyArray(object id)

{

int threadId = (int)id;

for (int i = 0; i < 10; i++)

{

arr[i] += threadId;

Thread.Sleep(70);

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (!int.TryParse(textBox1.Text, out N))

{

MessageBox.Show("Invalid number of threads.");

return;

}

textBox2.AppendText("Початковий стан масиву: ");

for (int i = 0; i < 10; i++)

textBox2.AppendText(arr[i] + " ");

textBox2.AppendText(Environment.NewLine);

Task[] tasks = new Task[N];

for (int i = 0; i < N; i++)

{

int id = i + 1;

tasks[i] = Task.Run(() => ModifyArray(id));

}

Task.WaitAll(tasks);

textBox2.AppendText("Модифікований стан масиву: ");

for (int i = 0; i < 10; i++)

{

textBox2.AppendText(arr[i] + " ");

Thread.Sleep(70);

}

textBox2.AppendText(Environment.NewLine);

}

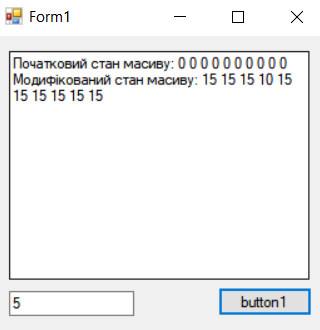
private void Form1\_Load(object sender, EventArgs e)

{

}

}

}



*Лістинг 3.5.1*

using System;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Lab3\_t3\_5\_1

{

public partial class Form1 : Form

{

private int[] arr = new int[10];

private int N;

public Form1()

{

InitializeComponent();

button1.Click += button1\_Click;

}

private void Run()

{

for (int i = 0; i < 10; i++)

{

arr[i] = i + 1;

Thread.Sleep(70);

}

}

private void ModifyArray(object id)

{

int threadId = (int)id;

for (int i = 0; i < 10; i++)

{

arr[i] += threadId;

Thread.Sleep(70);

}

}

private void button1\_Click(object sender, EventArgs e)

{

if (!int.TryParse(textBox1.Text, out N))

{

MessageBox.Show("Invalid number of threads.");

return;

}

textBox2.AppendText("Початковий стан масиву: ");

for (int i = 0; i < 10; i++)

textBox2.AppendText(arr[i] + " ");

textBox2.AppendText(Environment.NewLine);

Task[] tasks = new Task[N];

for (int i = 0; i < N; i++)

{

int id = i + 1;

tasks[i] = Task.Run(() => ModifyArray(id));

}

Task.WaitAll(tasks);

textBox2.AppendText("Модифікований стан масиву: ");

for (int i = 0; i < 10; i++)

{

textBox2.AppendText(arr[i] + " ");

Thread.Sleep(70);

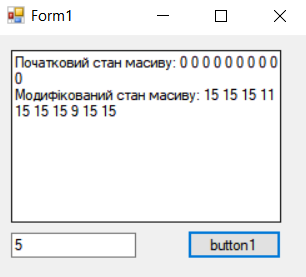
}

textBox2.AppendText(Environment.NewLine);

}

}

}



1. 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.

using System;

using System.Drawing;

using System.Threading;

using System.Windows.Forms;

namespace Lab3\_t4\_v5

{

public partial class Form1 : Form

{

private Point mousePosition;

private Point circlePosition;

private readonly int circleRadius = 50;

public Form1()

{

InitializeComponent();

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

MouseMove += Form1\_MouseMove;

Thread thread = new Thread(DrawCircle);

thread.IsBackground = true;

thread.Start();

}

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

mousePosition = e.Location;

}

private void DrawCircle()

{

while (true)

{

using (Graphics g = CreateGraphics())

{

g.Clear(BackColor);

Rectangle circleBounds = new Rectangle(circlePosition.X - circleRadius, circlePosition.Y - circleRadius, 2 \* circleRadius, 2 \* circleRadius);

g.FillEllipse(Brushes.Aqua, circleBounds);

if (Math.Abs(mousePosition.X - circlePosition.X) < circleRadius && Math.Abs(mousePosition.Y - circlePosition.Y) < circleRadius)

{

int dx = mousePosition.X - circlePosition.X;

int dy = mousePosition.Y - circlePosition.Y;

circlePosition.X -= dx;

circlePosition.Y -= dy;

}

}

Thread.Sleep(50);

}

}

}

}

