**Міністерство освіти України**

**Національний технічний університет України**

**“Київський політехнічний інститут імені Ігоря Сікорського”**

**Факультет інформатики та обчислювальної техніки**

**ЗВІТ**

до лабораторної роботи № 1

з дисципліни “Технології паралельних та розподілених обчислень”

Виконала:

Студентка групи ІП-71

Каспрук Анастасія Андріївна

Київ 2020

**Програмний код:**

1. КласMatrix

using System;

using System.Collections.Generic;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace Lab1

{

class Matrix

{

private readonly int \_dimension;

private readonly int[,] \_matrix;

public Matrix(int dimension)

{

\_dimension = dimension;

\_matrix = new int[dimension, dimension];

var random = new Random();

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

{

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

{

if (i == j) continue;

\_matrix[i, j] = random.Next(0, 100);

}

}

}

public Matrix(int[,] squareMatrix)

{

if(squareMatrix == null)

throw new ArgumentNullException(nameof(squareMatrix));

if (squareMatrix.Rank != 2 || squareMatrix.GetLength(0) != squareMatrix.GetLength(1))

throw new ArgumentException("Matrix is not square matrix");

\_matrix = squareMatrix;

}

public override string ToString()

{

var strBld = new StringBuilder();

for (int i = 0; i < \_dimension; i++)

{

for (int j = 0; j < \_dimension; j++)

{

strBld.Append($"{\_matrix[i, j]}\t");

}

strBld.AppendLine();

}

return strBld.ToString();

}

// One-threaded execution

public void CalcMainDiagonal()

{

for (int i = 0; i < \_dimension; i++)

{

CalcDiagonalValue(i);

}

}

// Multi-threaded execution (old style)

public void CalcMainDiagonalAsParallelOldFashioned()

{

List<Thread> threads = new List<Thread>(\_dimension);

Thread thread;

for (int i = 0; i < \_dimension; i++)

{

int loc = i;

thread = new Thread(() => CalcDiagonalValue(loc));

thread.Start();

threads.Add(thread);

}

threads.ForEach(t => t.Join());

}

// Multi-threaded execution (new style)

public async Task CalcMainDiagonalAsParallelNewStyleAsync()

{

List<Task> tasks = new List<Task>(\_dimension);

for (int i = 0; i < \_dimension; i++)

{

int loc = i;

tasks.Add(Task.Factory.StartNew(() => CalcDiagonalValue(loc)));

}

await Task.WhenAll(tasks);

}

private void CalcDiagonalValue(int index)

{

for (int j = 0; j < \_dimension; j++)

{

if (index == j) continue;

\_matrix[index, index] += \_matrix[index, j];

\_matrix[index, index] += \_matrix[j, index];

}

}

}

}

1. Клас Program

using System;

using System.Diagnostics;

namespace Lab1

{

class Program

{

static void Main(string[] args)

{

int dimension = 20000;

var matrix = new Matrix(dimension);

var swOneThread = Stopwatch.StartNew();

matrix.CalcMainDiagonal();

swOneThread.Stop();

Console.WriteLine(

$"One thread matrix main diagonal calculation time: {swOneThread.ElapsedMilliseconds} ms");

Console.WriteLine();

var swMultipleThreads = Stopwatch.StartNew();

matrix.CalcMainDiagonalAsParallelOldFashioned();

swMultipleThreads.Stop();

Console.WriteLine(

$"Multiple threads matrix main diagonal calculation time (old fashioned parallel programming style): {swMultipleThreads.ElapsedMilliseconds} ms");

double accelerationFactorOld = (double)swOneThread.ElapsedMilliseconds / swMultipleThreads.ElapsedMilliseconds;

Console.WriteLine(

$"Acceleration factor (Коефiцiєнт прискорення) for old fashioned parallel programming style: {accelerationFactorOld}");

Console.WriteLine($"Algorithm efficiency (Ефективнiсть алгоритму): {accelerationFactorOld / dimension}");

Console.WriteLine();

var swMultipleThreadsTask = Stopwatch.StartNew();

matrix.CalcMainDiagonalAsParallelNewStyleAsync().Wait();

swMultipleThreadsTask.Stop();

Console.WriteLine(

$"Multiple threads matrix main diagonal calculation time (new style of parallel programming): {swMultipleThreadsTask.ElapsedMilliseconds} ms");

double accelerationFactorNew = (double)swOneThread.ElapsedMilliseconds / swMultipleThreadsTask.ElapsedMilliseconds;

Console.WriteLine(

$"Acceleration factor (Коефiцiєнт прискорення) for new style of parallel programming: {accelerationFactorNew}");

Console.WriteLine($"Algorithm efficiency (Ефективнiсть алгоритму): {accelerationFactorNew / dimension}");

Console.WriteLine();

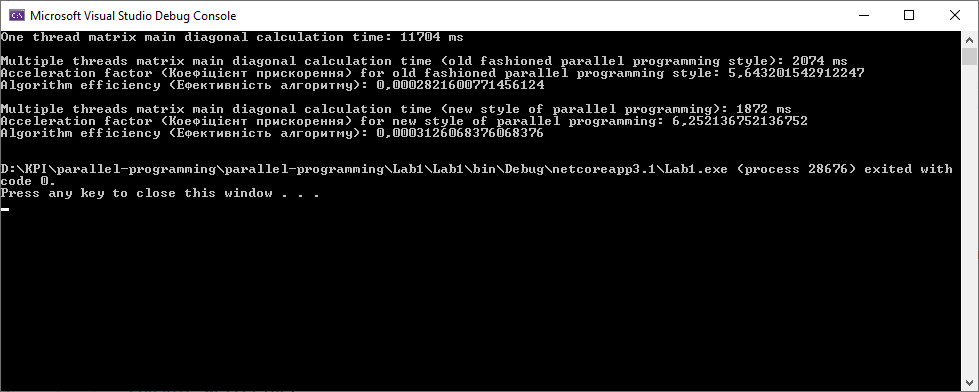
}

}

}

**Результат виконання:**

Для матриці розмірністю 20000.



Час виконання у одному потоці: 11704 мс.

Час виконання для класичного стилю роботи з декількома потоками: 2074 мс.

Коефіцієнт прискорення для класичного стилю роботи з потоками: 5,643201542912247.

Коефіцієнт ефективності для класичного стилю роботи з потоками: 0,0002821600771456124.

Час виконання для нового стилю роботи з декількома потоками: 1872 мс.

Коефіцієнт прискорення для нового стилю роботи з потоками: 6,252136752136752.

Коефіцієнт ефективності для нового стилю роботи з потоками: 0,0003126068376068376.