#### Министерство науки и высшего образования Российской Федерации

#### Федеральное государственное бюджетное образовательное учреждение

#### высшего профессионального образования

#### «Владимирский государственный университет

#### имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

Кафедра информационных систем и программной инженерии

Лабораторная работа №10

по дисциплине

«Технологии программирования»

Выполнил:

студент ИСТ-122

Г.Д. Симаненков

Принял:

ст. пр. каф. ИСПИ

В.В. Данилов

Владимир 2023 г.

**Цель работы**

Познакомиться с возможностями языка программирования C# при разработке многопоточных приложений.

**Выполнение работы**

Задание 2.

Реализовать процедуру сортировки большого массива без использования потоков и сделать тоже самое с использованием потоков. Времена выполнения сортировки для каждого из случаев фиксировать и построить график зависимости скорости сортировки от, например, числа потоков, объема данный и т.д.

Листинг программы.

namespace Sorting

{

class Program

{

static void Main()

{

int[] array = GenerateArray(10000000);

for (int i = 1; i <= Environment.ProcessorCount; i++)

{

int[] testArray = new int[array.Length];

Array.Copy(array, testArray, testArray.Length);

Console.Write(Sort.ParallelQuickSort(testArray, i) + " ");

}

Console.WriteLine();

for (int i = 1; i <= Environment.ProcessorCount; i++)

{

int[] testArray = new int[array.Length];

Array.Copy(array, testArray, testArray.Length);

Console.Write(Sort.QuickSort(testArray) + " ");

}

}

static int[] GenerateArray(int length)

{

Random random = new Random();

int[] array = new int[length];

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

{

array[i] = random.Next(1, 100);

}

return array;

}

}

public class Sort

{

public static long QuickSort(int[] array)

{

Stopwatch sw = Stopwatch.StartNew();

int length = array.Length;

sw.Start();

\_QuickSort(array, 0, length - 1);

sw.Stop();

return sw.ElapsedMilliseconds;

}

private static void \_QuickSort(int[] array, int first, int last)

{

if (first >= last)

{

return;

}

int middle = array[(first + last) / 2];

int left = first, right = last;

while (left < right)

{

while (array[left] < middle)

{

left++;

}

while (array[right] > middle)

{

right--;

}

if (left <= right)

{

int temp = array[left];

array[left] = array[right];

array[right] = temp;

left++;

right--;

}

}

\_QuickSort(array, first, right);

\_QuickSort(array, left, last);

}

public static long ParallelQuickSort(int[] array, int threadsCount)

{

int numThreads = Math.Max(1, Math.Min(threadsCount, Environment.ProcessorCount));

Stopwatch sw = Stopwatch.StartNew();

int length = array.Length;

sw.Start();

\_ParallelQuickSort(array, 0, length - 1, numThreads);

sw.Stop();

return sw.ElapsedMilliseconds;

}

private static void \_ParallelQuickSort(int[] array, int first, int last, int numThreads)

{

if (first >= last)

{

return;

}

int middle = array[(first + last) / 2];

int left = first, right = last;

while (left < right)

{

while (array[left] < middle)

{

left++;

}

while (array[right] > middle)

{

right--;

}

if (left <= right)

{

int temp = array[left];

array[left] = array[right];

array[right] = temp;

left++;

right--;

}

}

if (numThreads > 1)

{

int leftThreads = numThreads / 2;

int rightThreads = numThreads - leftThreads;

Parallel.Invoke(

() => \_ParallelQuickSort(array, first, right, leftThreads),

() => \_ParallelQuickSort(array, left, last, rightThreads));

}

else

{

\_QuickSort(array, first, right);

\_QuickSort(array, left, last);

}

}

}

}



Рисунок 3. Результат работы программы.

Рисунок 4. График зависимости времени от количества потоков.

**Вывод**

В ходе лабораторной работы были изучены возможности языка программирования C# при разработке многопоточных приложений.