**Параллельное программирование**

**Примеры программ**

**1.** Написать программу, находящую среднее значение элементов массива в 2-х потоках. (Идея решения: каждый поток суммирует свою часть массива, затем результаты объединяются.)

using System;

using System.Linq;

using System.Threading; // Содержит классы для работы с потоками

namespace Parallel

{

class Program

{

class Range {public int a, b;} // Класс для хранения диапазона суммируемых

// потоком элементов

static int[] arr; // Массив, элементы которого будут суммироваться

static int S = 0; // Сумма элементов массива

static object m = new object(); // Мьютекс для блокировки доступа к переменной S

// (object, но может быть любого ссылочного типа)

static void Sum(object range\_o) // Функция, выполняемая потоком, должна быть либо

{ // вида void f(), либо void f(object)

// Через range передаём диапазон номеров элементов, который будем суммировать

Range range = range\_o as Range;

int sum = 0; // Локальная сумма

for (int i = range.a; i < range.b; i++)

sum += arr[i];

// Синхронизация потоков

lock(m) // Так как S — общий ресурс, блокируем потоки на время доступа к нему

{

S += sum;

}

}

public static void Main()

{

// Создаём массив

const int N = 1000;

// N последовательных целых чисел, начиная с 1

arr = Enumerable.Range(1, N).ToArray();

Thread t1 = new Thread(Sum); // Создаем поток (нить), связанный с функцией Sum

Thread t2 = new Thread(Sum);

int M = arr.Length;

t1.Start(new Range{a = 0, b = M/2}); // Запуск потока с передачей параметров

t2.Start(new Range{a = M/2, b = M}); // Запуск второго потока

t1.Join(); // Дождаться завершения первого потока

t2.Join(); // Второго

Console.WriteLine("Среднее равно: {0}", (double)S / M); // Ответ: 500,5

}

}

}

или

using System;

using System.Linq;

using System.Threading; // Содержит классы для работы с потоками

namespace Parallel

{

class Program

{

static int[] arr; // Массив, элементы которого будут суммироваться

static int S = 0; // Сумма элементов массива

static object m = new object(); // Мьютекс для блокировки доступа к переменной S

// (object, но может быть любого ссылочного типа)

static void Sum(object range\_o) // Функция, выполняемая потоком должна быть либо

{ // вида void f(), либо void f(object)

// Через range передаём диапазон номеров элементов, который будем суммировать

int[] range = range\_o as int[];

int sum = 0; // Локальная сумма

for (int i = range[0]; i < range[1]; i++)

sum += arr[i];

lock (m) // Так как S — общий ресурс, блокируем потоки на время доступа к нему

{

S += sum;

}

}

public static void Main()

{

// Создаём массив

const int N = 1000;

// N последовательных целых чисел, начиная с 1

arr = Enumerable.Range(1, N).ToArray();

Thread t1 = new Thread(Sum); // Поток (нить), связанный с функцией Sum

Thread t2 = new Thread(Sum);

int M = arr.Length;

t1.Start(new int[] { 0, M / 2 }); // Запуск потока с передачей параметров

t2.Start(new int[] { M / 2, M }); // Запуск второго потока

t1.Join(); // Дождаться завершения первого потока

t2.Join(); // Второго

Console.WriteLine("Среднее равно: {0}", (double)S / M); // Ответ: 500,5

Console.ReadLine();

}

}

}

**2.** Написать программу, находящую среднее значение косинусов чётных элементов массива.

Измерить время выполнения в параллельном и непараллельном случаях.

using System;

using System.Linq;

using System.Diagnostics; // Для измерения времени

namespace Parallel

{

class Program

{

public static void Main()

{

Console.Write("Создаём массив...");

const int N = 10000000;

// N последовательных целых чисел, начиная с 1

int[] arr = Enumerable.Range(1, N).ToArray();

Console.WriteLine("OK");

Stopwatch sw = new Stopwatch(); // Объект для измерения времени (секундомер)

sw.Reset();

sw.Start();

var avg = (from x in arr

where x % 2 == 0

select Math.Cos(x)).Average();

sw.Stop();

Console.WriteLine("Среднее (последовательно): {0} ({1} мс)", avg, sw.ElapsedMilliseconds);

sw.Reset();

sw.Start();

var avg\_p = (from x in arr.AsParallel()

where x % 2 == 0

select Math.Cos(x)).Average();

sw.Stop();

Console.WriteLine("Среднее (параллельно): {0} ({1} мс)", avg\_p, sw.ElapsedMilliseconds);

}

}

}

**3.** Написать программу, находящую сумму элементов массива в 2-х потоках с помощью библиотеки TPL.

using System;

using System.Linq;

using System.Threading.Tasks;

namespace tasks

{

class Program

{

static int[] arr;

static int S = 0; // Глобальная сумма

static object m = new object(); // Мьютекс. Можно использовать

// любой объект ссылочного типа,

// даже уже существующий. Если

// блокируемый объект сам ссылочного

// типа, то можно его использовать в lock

class Range { public int a, b; }

static void Sum(object range\_o)

{

Range range = range\_o as Range;

int sum = 0;

for (int i = range.a; i < range.b; i++)

sum += arr[i];

lock (m)

{

S += sum;

}

}

static void Main(string[] args)

{

const int N = 10000;

arr = Enumerable.Range(1, N).ToArray();

int M = arr.Length;

var t1 = new Task(Sum, new Range() {a=0, b=M/2});

// Создание задачи

// Для задачи не гарантируется, что

// она будет запущена в отдельном потоке.

// Если задач много, то, возможно, часть из них

// будет запущена позже, чтобы не создавать сразу

// много потоков.

// Задача запускается быстрее, чем поток, так как

// она работает в потоке из т.н. пула потоков (заранее

// запущенного множества потоков, ждущих задания).

var t2 = new Task(Sum, new Range() { a = M / 2, b = M });

t1.Start();

t2.Start();

// Либо создание и одновременный запуск:

// Task.Factory.StartNew(Sum, new Range() { a = 0, b = M / 2 });

t1.Wait();

t2.Wait();

// Либо (удобно, если есть массив задач):

// Task.WaitAll(new[] { t1, t2 });

Console.WriteLine("S = {0}", S);

}

}

}

**Интернет-ресурсы**

1. <https://metanit.com/sharp/tutorial/>

Глава 15. Многопоточность (<https://metanit.com/sharp/tutorial/11.1.php>)

Глава 16. Параллельное программирование и библиотека TPL (<https://metanit.com/sharp/tutorial/12.1.php>)

Глава 21. Parallel LINQ (<https://metanit.com/sharp/tutorial/17.1.php>)