МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное учреждение высшего образования
«Чувашский государственный университет И.Н. Ульянова»
Факультет информатики и вычислительной техники
Кафедра вычислительной техники

Параллельное программирование
Лабораторная работа 2
Выполнение заданий с 6 - 12

Выполнил:

Студент группы ИВТ-41-20
Галкин Д.С.

Проверил:

Ковалев С.В.

Чебоксары, 2024

Цель работы (Технология программирования в System.Threading.Tasks):

Задание 6. Распараллеливание циклов в System.Threading.Tasks (параметр schedule):

> **Задание для выполнения**
>
> Изучите параметр schedule директивы for. Модифицируйте программу "Сумма чисел" из задания 5 таким образом, чтобы дополнительно выводилось на экран сообщение о том, какая нить какую итерацию выполняет:
> [<Номер нити>]: calculation of the iteration number [<Номер итерации>].
>
> Задайте k = 4, N = 10. Заполните следующую таблицу распределения итераций цикла по нитям в зависимости от параметра schedule

Полный текст программы:

1. Класс Main

```
var lab6 = new Lab6("Lab6");
lab6.Start();
```

2. Класс Lab6

```
using System.Collections.Concurrent;
using ParallelsProgramming.Intrefeces;

namespace ParallelsProgramming.Labs;

public class Lab6 : ILab
{
    public string Name { get; set; }
    private const int K = 4; // Количество нитей
    private const int N = 10; // Количество итераций
    private const int chunkSize = 2; // Размер блока для static schedule
    public Lab6(string name)
    {        Name = name;
    }    public void Start()
    {        Console.WriteLine(this);
        ThreadStart();
        Console.WriteLine("\n");
    }    public void ThreadStart()
    {        StaticSchedule();
        DynamicSchedule();
    }
    public override string ToString()
    {        return $"{Name} started classes:";
    }

    private void StaticSchedule()
    {        // Static Schedule
        Console.WriteLine("Static Schedule:");
        for (int i = 0; i < N; i += chunkSize)
        {            Parallel.For(i, Math.Min(i + chunkSize, N), Print);
        }    }    private void DynamicSchedule()
    {        // Создание списка задач
        var tasks = new Task[K];
        // Очередь для хранения номеров итераций
        var iterationsQueue = new ConcurrentQueue<int>();
        for (int i = 0; i < N; i++)
        {            iterationsQueue.Enqueue(i);
        }        Console.WriteLine("Dynamic Schedule:");
        for (int i = 0; i < K; i++)
        {            tasks[i] = Task.Run(() =>
            {
                while (iterationsQueue.TryDequeue(out int iteration))
                {                    Print(iteration + 1);

                    // Имитация динамического распределения с chunk
                    for (int j = 1; j < chunkSize && iterationsQueue.TryDequeue(out
int nextIteration); j++)
                    {                        Print(nextIteration + 1);
                    }                }            });        }
        Task.WaitAll(tasks); // Ожидание завершения всех задач
    }

    private void Print(int index) => Console.WriteLine($"[Thread {Task.CurrentId}]:
```

```
calculation of the iteration number {index + 1}.");
}
```

Результат:

```
Lab6 started classes:
Static Schedule:
[Thread 2]: calculation of the iteration number 2.
[Thread 1]: calculation of the iteration number 1.
[Thread 3]: calculation of the iteration number 3.
[Thread 3]: calculation of the iteration number 4.
[Thread 4]: calculation of the iteration number 5.
[Thread 4]: calculation of the iteration number 6.
[Thread 5]: calculation of the iteration number 7.
[Thread 5]: calculation of the iteration number 8.
[Thread 6]: calculation of the iteration number 9.
[Thread 6]: calculation of the iteration number 10.
Dynamic Schedule:
[Thread 7]: calculation of the iteration number 2.
[Thread 7]: calculation of the iteration number 6.
[Thread 7]: calculation of the iteration number 7.
[Thread 9]: calculation of the iteration number 4.
[Thread 7]: calculation of the iteration number 8.
[Thread 9]: calculation of the iteration number 9.
[Thread 7]: calculation of the iteration number 10.
[Thread 9]: calculation of the iteration number 11.
[Thread 8]: calculation of the iteration number 3.
[Thread 10]: calculation of the iteration number 5.
```

Задание 7. Распараллеливание циклов в System.Threading.Tasks (программа "Число pi"):

> ⓘ **Задание для выполнения**
>
> Напишите программу, которая вычисляет число pi с точностью до N знаков после запятой. Используйте следующую формулу:

$$\pi = \left(\frac{4}{1+x_0^2} + \frac{4}{1+x_1^2} + \ldots + \frac{4}{1+x_{N-1}^2} \right) \times \frac{1}{N}, \text{ где } x_i = (i+0.5) \times \frac{1}{N}, \; i = \overline{0, N-1}$$

Входные данные: Одно целое число N (точность вычисления)
Выходные данные: одно вещественное число pi

Входные данные	Выходные данные
1000000000	3.14159265

Полный текст программы:

1. Класс Main

```
var lab7 = new Lab7("Lab7");
lab7.Start();
```

2. Класс Lab7

```
using ParallelsProgramming.Intrefeces;

namespace ParallelsProgramming.Labs;

public class Lab7 : ILab
{
    public string Name { get; set; }
    public Lab7(string name)
    {        Name = name;
    }    public void Start()
    {        Console.WriteLine(this);
        ThreadStart();
        Console.WriteLine("\n");
    }    public void ThreadStart()
    {        Console.WriteLine("Введите целое число N: ");
        if (!int.TryParse(Console.ReadLine(), out int N) && N <= 0)
        {            Console.WriteLine("Некорректный ввод. Убедитесь, что вводите
положительное целое число.");
            return;
        }
var pi = GetNumberPi(N);

        Print(pi);
    }
    public override string ToString()
    {        return $"{Name} started classes:";
    }

    private void Print(double pi) => Console.WriteLine($"Result PI = {pi}");

    private double GetNumberPi(int n)
    {
        double sum = 0.0;
        double step = 1.0 / n;

        object locker = new();
        Parallel.For(0, n - 1, i =>
        {
            double x = (i + 0.5) * step;
            double term = 4.0 / (1.0 + x * x);
            lock (locker)
            {                sum += term;            }        });
        return sum * step;
    }    }
```

Результат:

```
Lab7 started classes:
Введите целое число N:
1000000000
Result PI = 3,1415926515898476
```

Задание 8. Распараллеливание циклов в System.Threading.Tasks (программа "Матрица"):

Задание для выполнения

Напишите программу, которая вычисляет произведение двух квадратных матриц AxB = C размера n * n. Используйте формулу:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ . & . & . & \dots & . \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & \dots & b_{1n} \\ b_{21} & b_{22} & b_{23} & \dots & b_{2n} \\ . & . & . & \dots & . \\ b_{n1} & b_{n2} & b_{n3} & \dots & b_{nn} \end{pmatrix}$$

$$c_{im} = \sum_{j=1}^{n} a_{ij} \cdot b_{jm}; i = 1,2,\dots,n;\ m = 1,2,\dots,n$$

$$C = \begin{pmatrix} \sum_{j=1}^{n} a_{1j} \cdot b_{j1} & \sum_{j=1}^{n} a_{1j} \cdot b_{j2} & \sum_{j=1}^{n} a_{1j} \cdot b_{j3} & \dots & \sum_{j=1}^{n} a_{1j} \cdot b_{jn} \\ \sum_{j=1}^{n} a_{2j} \cdot b_{j1} & \sum_{j=1}^{n} a_{2j} \cdot b_{j2} & \sum_{j=1}^{n} a_{2j} \cdot b_{j3} & \dots & \sum_{j=1}^{n} a_{2j} \cdot b_{jn} \\ \dots & \dots & \dots & \dots & \dots \\ \sum_{j=1}^{n} a_{nj} \cdot b_{j1} & \sum_{j=1}^{n} a_{nj} \cdot b_{j2} & \sum_{j=1}^{n} a_{nj} \cdot b_{j3} & \dots & \sum_{j=1}^{n} a_{nj} \cdot b_{jn} \end{pmatrix}$$

Входные данные: целое число n, 1<=n<=10, n^2 вещественных элементов матрицы A и n^2 вещественных элементов матрицы B.
Выходные данные: n^2 вещественных элементов матрицы C

Входные данные	Выходные данные
2 1 3 4 8 5 4 3 0	14 4 44 16

Полный текст программы:

1. Класс Main

```
var lab8 = new Lab8("Lab8");
lab8.Start();
```

2. Класс Lab8

```
using ParallelsProgramming.Intrefeces;

namespace ParallelsProgramming.Labs;

public class Lab8 : ILab
{
    public string Name { get; set; }
    public Lab8(string name)
    {        Name = name;
    }    public void Start()
    {        Console.WriteLine(this);
        ThreadStart();
        Console.WriteLine("\n");
    }    public void ThreadStart()
    {        Console.WriteLine("Введите размер матриц n x n: ");
        int n = int.Parse(Console.ReadLine() ?? "0");
        Console.WriteLine("Заполнение матрицы A:");
        int[,] A = ReadMatrix(n);

        Console.WriteLine("Заполнение матрицы B:");
        int[,] B = ReadMatrix(n);
        // Умножение матриц
        int[,] C = MultiplyMatricesParallel(A, B, n);

        // Вывод результата
        Console.WriteLine("Матрица C (A x B):");
        PrintMatrix(C, n);
    }
    public override string ToString()
    {        return $"{Name} started classes:";
    }
private int[,] ReadMatrix(int n)
    {        var matrix = new int[n, n];
        for (int i = 0; i < n; i++)
        {            for (int j = 0; j < n; j++)
            {                Console.Write($"Введите элемент [{i + 1}, {j + 1}]: ");
                matrix[i, j] = int.Parse(Console.ReadLine() ?? "0"); // Примерные
значения элементов матрицы
            }
        }        return matrix;
    }    private int[,] MultiplyMatricesParallel(int[,] A, int[,] B, int n)
    {        var C = new int[n, n];

        Parallel.For(0, n, i =>
        {
            for (int j = 0; j < n; j++)
            {                C[i, j] = 0;
                for (int k = 0; k < n; k++)
                {                    C[i, j] += A[i, k] * B[k, j];                }
}        });
        return C;
    }    private void PrintMatrix(int[,] matrix, int n)
    {        for (int i = 0; i < n; i++)
        {            for (int j = 0; j < n; j++)
```

```
            {                Console.Write($"{matrix[i, j]} ");
            }            Console.WriteLine();
        }    }}
```

Результат:

```
Lab8 started classes:
Введите размер матриц n x n:
2
Заполнение матрицы A:
Введите элемент [1, 1]: 1
Введите элемент [1, 2]: 3
Введите элемент [2, 1]: 4
Введите элемент [2, 2]: 8
Заполнение матрицы B:
Введите элемент [1, 1]: 5
Введите элемент [1, 2]: 4
Введите элемент [2, 1]: 3
Введите элемент [2, 2]: 0
Матрица C (A x B):
14 4
44 16
```

Задание 9. Параллельные секции в System.Threading.Tasks (программа "I'm here"):

> ⓘ **Задание для выполнения**
>
> Изучите директивы создания параллельных секций sections и section. Напишите программу, содержащую 3 параллельные секции, внутри каждой из которых должно выводиться сообщение:
>
> [<Номер нити>]: came in section <Номер секции>
>
> Вне секций внутри параллельной области должно выводиться следующее сообщение:
>
> [<Номер нити>]: parallel region
>
> Запустите приложене на 2-х, 3-х, 4-х нитях. Проследите, как нити распределяются по параллельным секциям

Входные данные: k - кол-во нитей в параллельной области
Выходные данные: k строка вида "[<Номер нити>]: come in section <Номер секции>", k-строка вида "[<Номер нити>]: parallel region"

Входные данные	Выходные данные
3	1: came in section 1 2: came in section 2 3: came in section 3 1: parallel region 2: parallel region 3: parallel region

Полный текст программы:

1. Класс Main

```
var lab9 = new Lab9("Lab9");
lab9.Start();
```

2. Класс Lab9

```
using ParallelsProgramming.Intrefeces;

namespace ParallelsProgramming.Labs;

public class Lab9 : ILab
{
    public string Name { get; set; }

    public Lab9(string name)
    {        Name = name;
    }
    public void ThreadStart()
    {        Parallel.Invoke(
            () => ParallelRegion(1),
            () => ParallelRegion(2),
            () => ParallelRegion(3)
        );    }
    public void Start()
    {        Console.WriteLine(this);
        ThreadStart();
        Console.WriteLine("\n");
    }
    public override string ToString()
    {        return $"{Name} started classes:";
    }

    private void ParallelRegion(int sectionNumber)
    {        Console.WriteLine($"[{Task.CurrentId}]: came in section
{sectionNumber}");
        Parallel.For(0, 1, i =>
        {
            Console.WriteLine($"[{Task.CurrentId}]: parallel region");
        });    }}
```

Результат:

```
Lab9 started classes:
[1]: came in section 1
[2]: came in section 2
[3]: came in section 3
[4]: parallel region
[5]: parallel region
[6]: parallel region
```

Задание 10. Гонка потоков System.Threading.Tasks (программа "Сумма чисел" с atomic)":

> **Задание для выполнения**
>
> Перепишите программу, в которой параллельно вычисляется сумма чисел от 1 до N, без использования параметра reduction. Вместо параметра reduction используйте директиву atomic

Входные данные: целое число N - количество чисел
Выходные данные: каждая нить выводит всю частичную сумму в формате "[Номер нити]: Sum = [Частичная сумма]", один раз выводится общая сумма в формате "Sum = [Сумма]"

Полный текст программы:

1. Класс Main

```
var lab10 = new Lab10("Lab10");
lab10.Start();
```

2. Класс Lab10

```
using ParallelsProgramming.Intrefeces;

namespace ParallelsProgramming.Labs;

public class Lab10 : ILab
{
    public string Name { get; set; }
    private readonly object _lockObject = new object(); // Объект блокировки
    public Lab10(string name)
    {        Name = name;
    }    public void ThreadStart()
    {        Console.WriteLine("Введите кол-во нитей K: ");
        if (!int.TryParse(Console.ReadLine(), out int K) && K <= 0)
        {            Console.WriteLine("Некорректный ввод. Убедитесь, что вводите
положительное целое число.");
            return;
        }        Console.WriteLine("Введите число N: ");
        if (!int.TryParse(Console.ReadLine(), out int N) && N <= 0)
        {            Console.WriteLine("Некорректный ввод. Убедитесь, что вводите
положительное целое число.");
            return;
        }        int[] partialSums = new int[K];
        Task[] tasks = new Task[K];
        Parallel.For(0, K, new ParallelOptions { MaxDegreeOfParallelism = K }, taskNum
=>
        {
            var self = N > K ? N / K : K / N;
            // Определение диапазона для каждой нити
            int localTaskNum = taskNum; // Локальная переменная для каждой итерации
            int start = localTaskNum * (self) + 1;
            int end = (localTaskNum == K - 1) ? N : (localTaskNum + 1) * (self);

            // Вычисление частичной суммы для каждой нити
            for (int i = start; i <= end; i++)
            {                lock (_lockObject)
                {                    partialSums[taskNum] += i;                }
}
            Print(localTaskNum, partialSums[localTaskNum]);
        });
        int totalSum = 0;
        foreach (int sum in partialSums)
        {            totalSum += sum;        }
        Console.WriteLine($"Sum = {totalSum}"); // Вывод общей суммы
    }

    public void Start()
    {        Console.WriteLine(this);
        ThreadStart();
        Console.WriteLine("\n");
    }
    public override string ToString()
    {        return $"{Name} started classes:";
    }
```

```
    private void Print(int index, int sum) => Console.WriteLine($"[{index}]: Sum =
{sum}");
}
```

Результат:

```
Lab10 started classes:
Введите кол-во нитей K:
2
Введите число N:
4
[0]: Sum = 3
[1]: Sum = 7
Sum = 10
```

Задание 11. Гонка потоков System.Threading.Tasks (программа "Число pi" с critical)":

> **Задание для выполнения**
>
> Перепишите параллельную программу вычисления числа pi без использования параметра reduction. Вместо параметра reduction используйте директиву critical.

Полный текст программы:

1. Класс Main

```
var lab11 = new Lab10("Lab11");
lab11.Start();
```

2. Класс Lab11

```
using ParallelsProgramming.Intrefeces;

namespace ParallelsProgramming.Labs;

public class Lab11 : ILab
{
    public string Name { get; set; }
    public Lab11(string name)
    {        Name = name;
    }    public void Start()
    {        Console.WriteLine(this);
        ThreadStart();
        Console.WriteLine("\n");
    }    public void ThreadStart()
    {        Console.WriteLine("Введите целое число N: ");
        if (!int.TryParse(Console.ReadLine(), out int N) && N <= 0)
        {            Console.WriteLine("Некорректный ввод. Убедитесь, что вводите
положительное целое число.");
            return;
        }
var pi = GetNumberPi(N);

        Print(pi);
    }
    public override string ToString()
    {        return $"{Name} started classes:";
    }

    private void Print(double pi) => Console.WriteLine($"Result PI = {pi}");

    private double GetNumberPi(int n)
    {
        double sum = 0.0;
        double step = 1.0 / n;

        object locker = new();
        Parallel.For(0, n - 1, i =>
        {
            double x = (i + 0.5) * step;
            double term = 4.0 / (1.0 + x * x);
            double localSum = term * step; // Вычисление частичной суммы для итерации
            lock (locker)
            {                sum += localSum;            }        });
        return sum;
    }}
```

Результат:

```
Lab11 started classes:
Введите целое число N:
1000000000
Result PI = 3,1415926515896224
```