**14 МНОГОПОТОЧНЫЕ ПРИЛОЖЕНИЯ**

Задание 1. Создать консольное приложение, в котором реализовано 3 потока. Первый выводит числа от 0 до 9, второй – от 10 до 19, третий – от 20 до 29. Вывод значений на экран происходит следующим образом: сначала своё число выводит первый поток, а затем второй и третий. Организовать возможность использования методов Start и Sleep, а также изменение свойства Priority, позволяющее изменить приоритет потока. Можно воспользоваться инструментами синхронизации потоков lock, AutoResetEvent и т.д.

Листининг программы:

using System;

using System.Threading;

namespace task1

{

class Program

{

static object lockObj = new object();

static void FirstThread()

{

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

{

Thread.Sleep(100);

WriteLine($"Первый поток говорит: {i}",ConsoleColor.Red);

Thread.Sleep(100);

}

}

static void SecondThread()

{

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

{

Thread.Sleep(100);

WriteLine($"Второй поток говорит: {i}", ConsoleColor.Yellow);

Thread.Sleep(100);

}

}

static void ThirdThread()

{

for (int i = 20; i<30; i++)

{

Thread.Sleep(100);

WriteLine($"Третий поток говорит: {i}", ConsoleColor.Green);

Thread.Sleep(100);

}

}

static void Main(string[] args)

{

Thread thread1 = new Thread(new ThreadStart(FirstThread));

Thread thread2 = new Thread(new ThreadStart(SecondThread));

Thread thread3 = new Thread(new ThreadStart(ThirdThread));

thread1.Priority = ThreadPriority.Highest;

thread2.Priority = ThreadPriority.Normal;

thread3.Priority = ThreadPriority.Lowest;

thread1.Start();

Thread.Sleep(100);

thread2.Start();

Thread.Sleep(100);

thread3.Start();

}

static void WriteLine(string text, ConsoleColor color)

{

lock (lockObj)

{

Console.ForegroundColor = color;

Console.WriteLine(text);

Console.ForegroundColor = ConsoleColor.White;

}

}

}

}

Таблица 14.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Первый поток говорит: 0  Второй поток говорит: 10  Третий поток говорит: 20  Первый поток говорит: 1  Второй поток говорит: 11  Третий поток говорит: 21  Первый поток говорит: 2  Второй поток говорит: 12  Третий поток говорит: 22  Второй поток говорит: 13  Первый поток говорит: 3  Третий поток говорит: 23  Первый поток говорит: 4  Второй поток говорит: 14  Третий поток говорит: 24  Первый поток говорит: 5  Второй поток говорит: 15  Третий поток говорит: 25  Первый поток говорит: 6  Второй поток говорит: 16  Третий поток говорит: 26  Первый поток говорит: 7  Второй поток говорит: 17  Третий поток говорит: 27  Первый поток говорит: 8  Второй поток говорит: 18  Третий поток говорит: 28  Первый поток говорит: 9  Второй поток говорит: 19  Третий поток говорит: 29 |

Анализ результатов:

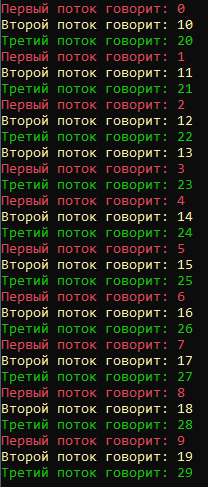


Рисунок 14.1 – Результат работы программы

Источник – собственная разработка

Задание 2. Создать консольное приложение, в котором несколько потоков будут выполнять один и тот же метод. (Количество потоков: 2, Метод: Сумма чисел от 1 до 10). Произвести расчет затраченного времени в миллисекундах на выполнение потока и вывести его на экран.

Листининг программы:

using System;

using System.Threading;

using System.Diagnostics;

namespace task2

{

class Program

{

static void Calculation()

{

Stopwatch sWatch = new Stopwatch();

sWatch.Start();

int sum = 0;

for (int i = 1; i < 11; i++)

{

sum += i;

Thread.Sleep(10);

}

Console.WriteLine(sum);

sWatch.Stop();

Console.WriteLine($"Время на выполнение {Thread.CurrentThread.Name}: {sWatch.ElapsedMilliseconds}");

}

static void Main(string[] args)

{

Thread thread1 = new Thread(new ThreadStart(Calculation));

thread1.Name = "поток1";

Thread thread2 = new Thread(new ThreadStart(Calculation));

thread2.Name = "поток2";

thread1.Start();

Thread.Sleep(100);

thread2.Start();

}

}

}

Таблица 14.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Сообщение №1  Сообщение №2  Сообщение №3 |

Анализ результатов:

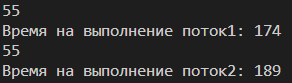


Рисунок 14.2 – Результат работы программы

Источник – собственная разработка

Задание 3. Создать консольное приложение, в котором будут два метода, один метод будет выполняться двумя потоками одновременно, а другой метод в каждый момент времени будет выполняться одним потоком. (Количество потоков: 2, Метод: 1. A+A 1 +A 2 +A 3 +..+A N , А и N вводятся с клавиатуры; 2. A\*A 1 \*A 2 \*A 3 \*…\*A N , А и N вводятся с клавиатуры ).

Листининг программы:

using System;

using System.Threading;

namespace task3

{

class Program

{

static void Plus()

{

Console.Write("Введите A: ");

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

Console.Write("Введите N: ");

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

double sum = 0;

for (int i = 1; i <= n; i++)

{

sum += Math.Pow(a, i);

}

Console.WriteLine(sum);

}

static void Multiplication()

{

Console.Write("Введите A: ");

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

Console.Write("Введите N: ");

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

double sum = 1;

for (int i = 1; i <= n; i++)

{

sum \*= Math.Pow(a, i);

}

Console.WriteLine(sum);

}

static void Main(string[] args)

{

Thread thread1 = new Thread(new ThreadStart(Plus));

Thread thread2 = new Thread(new ThreadStart(Plus));

thread1.Start();

thread2.Start();

Multiplication();

}

}

}

Таблица 14.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 20, 2, 2, 2, 2, 4 | 420, 6, 1024 |

Анализ результатов:

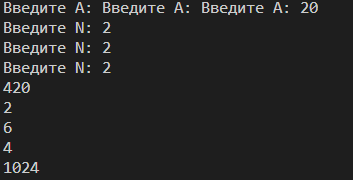


Рисунок 14.3 – Результат работы программы

Источник – собственная разработка

Задание 4. Дана последовательность натуральных чисел {a 0 …a n–1 }. Создать многопоточное приложение для поиска суммы ∑a i , где a i – четные числа.

Листининг программы:

using System;

using System.Threading;

namespace task4

{

class Program

{

public static int b = 10, sum = 0, CurrentThread = 0;

public static int[] c;

static void Main(string[] args)

{

int n = 1000;

int threads = 10;

c = new int[n];

Random r = new Random();

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

{

c[i] = r.Next(20);

}

Thread[] threadsArray = new Thread[threads];

for (int i = 0; i < threads - 1; i++)

{

threadsArray[i] = new Thread(new ThreadStart(delegate () {

for (int j = n / threads \* i; j < n / threads \* (i + 1); j++)

{

if (c[j] % 2 == 0)

{

sum += c[j];

}

}

}));

CurrentThread++;

threadsArray[i].Start();

threadsArray[i].Join();

}

Console.WriteLine("count:{0}", sum);

Console.ReadKey();

}

}

}

Таблица 14.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 3938 |

Анализ результатов:



Рисунок 14.4 – Результат работы программы

Источник – собственная разработка