**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.

Листинг программы:

static void FirstThread()

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

{ Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine(i);

Console.ForegroundColor = ConsoleColor.White;

Thread.Sleep(10);}}

static void SecondThread()

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

{ Console.ForegroundColor = ConsoleColor.Yellow;

Console.WriteLine(i);

Console.ForegroundColor = ConsoleColor.White;

Thread.Sleep(10);}}

static void ThirdThread()

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

{ Console.ForegroundColor = ConsoleColor.Green;

Console.WriteLine(i);

Console.ForegroundColor = ConsoleColor.White;

Thread.Sleep(10); }}

Thread thread = new Thread(FirstThread);

Thread thread1 = new Thread(SecondThread);

Thread thread2 = new Thread(ThirdThread);

thread.Priority = ThreadPriority.Highest;

thread1.Priority = ThreadPriority.Normal;

thread2.Priority = ThreadPriority.Lowest;

thread.Start();

thread.Join();

thread1.Start();

thread1.Join();

thread2.Start();

thread2.Join();

Таблица 14.1 – Входные и выходные данных задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 0 20 10 11 21 1 2 12 22 23 |

Анализ результатов:

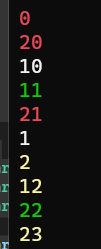


Рисунок 14.1 – Результат работы программы из задания №1

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.

Листинг программы:

using System.Threading;

using System.Diagnostics;

object locker = new object();

void Sum()

{ lock (locker)

{ Console.WriteLine(Thread.CurrentThread.Name);

int sum = 0;

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

{ sum += i; }

Console.WriteLine(sum);

Thread.Sleep(10); }}

Stopwatch sw = new Stopwatch();

Thread thread = new Thread(Sum);

thread.Name = "Thread";

Thread thread1 = new Thread(Sum);

thread1.Name = "Thread 1";

sw.Start();

thread.Start();

sw.Stop();

Console.WriteLine("Миллисекунды 1: "+sw.ElapsedMilliseconds.ToString());

sw.Stop();

thread1.Start();

sw.Stop();

Console.WriteLine("Миллисекунды 2: "+sw.ElapsedMilliseconds.ToString());

Таблица 14.2 – Входные и выходные данных задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Миллисекунды: 0 Поток: 55  Миллисекунды:0 Поток 2: 55 |

Анализ результатов:

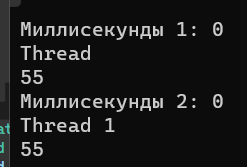


Рисунок 14.2 – Результат работы программы из задания №2

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+A1+A2+A3+..+AN, А и N вводятся с клавиатуры; 2. A\*A1\*A2\*A3\*…\*AN, А и N вводятся с клавиатуры ).

Листинг программы:

void ExponentiationInAddition(object obj)

{ if (obj is Params myparams)

{ double exp = 0;

for (double i = myparams.A; i < myparams.N; i++)

{ exp += Math.Pow(myparams.A, myparams.N); }

Console.WriteLine(exp);

Thread.Sleep(10); }}

object locker = new object();

void ExponentiationInMultiplication(double a, double n )

{ lock (locker)

{ double exp = 1;

for (double i = a; i < n; i++)

{ exp += exp \* Math.Pow(a, n); }

Console.WriteLine(exp);

Thread.Sleep(10);}}

Console.Write("Enter a: ");

double a = double.Parse(Console.ReadLine());

Console.Write("Enter n: ");

double n = double.Parse(Console.ReadLine());

Params myparams = new Params() { A = a, N = n };

Thread thread1 = new Thread(ExponentiationInAddition);

thread1.Name = "Thread 1";

Thread thread2 = new Thread(ExponentiationInAddition);

thread1.Name = "Thread 2";

Thread thread3 = new Thread(()=>ExponentiationInMultiplication(a, n));

thread3.Name = "Thread 1";

Thread thread4 = new Thread(()=>ExponentiationInMultiplication(a, n));

thread4.Name = "Thread 2";

thread1.Start(myparams);

thread1.Join();

thread2.Start(myparams);

thread2.Join();

thread3.Start();

thread4.Start();

Таблица 14.3 – Входные и выходные данных задание №3

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| A = 2  B=4 | 32 32  289 289 |

Анализ результатов:

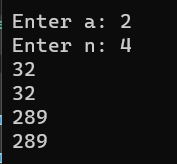


Рисунок 14.3 – Результат работы программы из задания №3