**13 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:

internal class Program

{

static void potok1()

{

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

{

Console.WriteLine("Поток 1 выводит: " + i);

}

}

static void potok2()

{

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

{

Console.WriteLine("Поток 2 выводит: " + i);

}

}

static void potok3()

{

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

{ Console.WriteLine("Поток 3 выводит: " + i); }

}

static void Main(string[] args)

{

Thread thread1 = new Thread(potok1);

Thread thread2 = new Thread(potok2);

Thread thread3 = new Thread(potok3);

thread1.Priority = ThreadPriority.Highest;

thread2.Priority = ThreadPriority.AboveNormal;

thread3.Priority = ThreadPriority.Lowest;

thread1.Start();

Thread.Sleep(1000);

thread2.Start();

Thread.Sleep(1000);

thread3.Start();

Console.ReadLine();

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Поток 1 выводит: 0  …  Поток 3 выводит: 29 |

Анализ результатов:

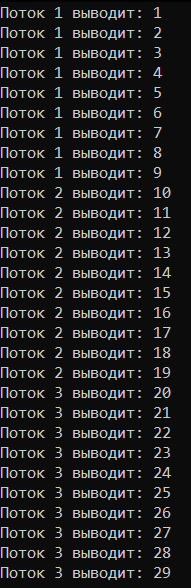


Рисунок 1.1 – Результат работы программы

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:

internal class Program

{

static void potok1()

{

int sum = 0;

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

{

sum += i;

Console.WriteLine("Поток 1 выводит: " + sum);

}

}

static void potok2()

{

int sum1 = 0;

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

{

sum1 += i;

Console.WriteLine("Поток 2 выводит: " + sum1);

}

}

static void Main(string[] args)

{

Stopwatch sWatch = new Stopwatch();

Thread thread1 = new Thread(potok1);

Thread thread2 = new Thread(potok2);

thread1.Priority = ThreadPriority.Highest;

thread2.Priority = ThreadPriority.AboveNormal;

sWatch.Start();

thread1.Start();

Thread.Sleep(1000);

sWatch.Stop();

thread2.Start();

Console.WriteLine("Время (в мс.) затраченное на первый поток = " + sWatch.ElapsedMilliseconds.ToString());

Console.ReadLine();

}

}

Таблица 1.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Поток 1 выводит 0  Поток 1 выводит 1  Поток 1 выводит 3  Поток 1 выводит 6  Поток 1 выводит 10  Поток 1 выводит 15  Поток 1 выводит 21  Поток 1 выводит 28  Поток 1 выводит 36  Поток 1 выводит 45  Поток 2 выводит 0  Поток 2 выводит 1  Поток 2 выводит 3  Поток 2 выводит 6  Поток 2 выводит 10  Поток 2 выводит 15  Поток 2 выводит 21  Поток 2 выводит 28  Поток 2 выводит 36  Поток 2 выводит 45  Время (в мс.) затраченное на первый поток = 1012 |

Анализ результатов:

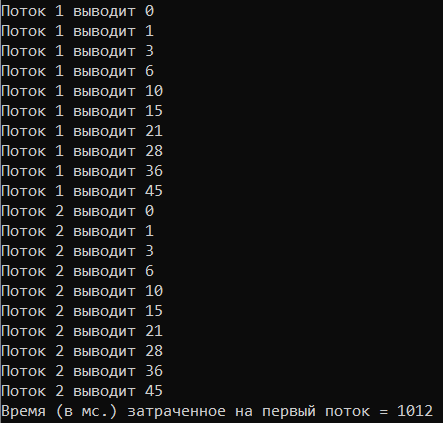


Рисунок 1.2 – Результат работы программы

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 ).

Листинг программы:

internal class Program

{

static void FirstMethod(int a)

{

double sum = 0;

for (int i = 1; i < 10; i++)

{

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

Console.WriteLine($"Первый метод: {sum}");

}

}

static void SecondMethod(double a)

{

double multiply = 1;

for (int j = 1; j < 10; j++)

{

multiply = multiply \* Math.Pow(a, j);

Console.WriteLine($"Второй метод: {multiply}");

}

}

static void Main(string[] args)

{

int a;

Console.Write("Введите А: ");

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

int n;

Console.Write("Введите N: ");

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

Thread thread1 = new Thread(() => FirstMethod(a));

Thread thread2 = new Thread(() => SecondMethod(n));

Thread thread3 = new Thread(() => FirstMethod(a));

thread1.Start();

thread2.Start();

thread3.Start();

Console.ReadLine();

}

}

Таблица 1.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1  3 | Первый метод: 1  Второй метод: 3  Первый метод: 1  Первый метод: 2  Первый метод: 3  Первый метод: 4  Первый метод: 5  Первый метод: 6  Первый метод: 7  Первый метод: 2  Первый метод: 3  Первый метод: 4  Второй метод: 27  Первый метод: 8  Первый метод: 9  Второй метод: 729  Первый метод: 5  Второй метод: 59049  Первый метод: 6  Первый метод: 7  Второй метод: 14348907  Второй метод: 10460353203  Первый метод: 8  Первый метод: 9  Второй метод: 22876792454961  Второй метод: 1,5009463529699914E+17  Второй метод: 2,954312706550834E+21 |

Анализ результатов:

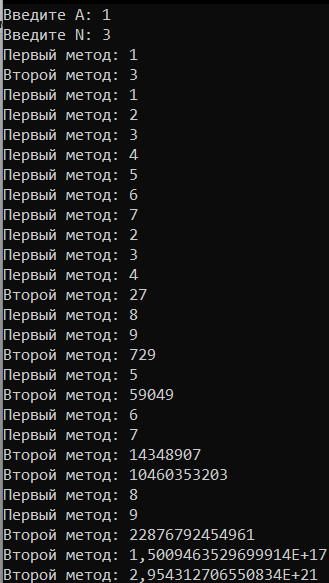


Рисунок 1.3 – Результат работы программы