**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.

Листинг программы:

class Program

{

static void FirstThread()

{

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

{

Console.WriteLine("Первый поток " + i);

}

Thread.Sleep(100);

}

static void SecondThread()

{

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

{

Console.WriteLine("Второй поток " + i);

}

Thread.Sleep(50);

}

static void ThirdThread()

{

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

{

Console.WriteLine("Третий поток " + i);

Thread.Sleep(10);

}

}

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;

thread3.Priority = ThreadPriority.Lowest;

thread1.Start();

thread2.Start();

thread3.Start();

}

}

Таблица 14.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Первый поток 0 | Третий поток 30 |

Анализ результатов:

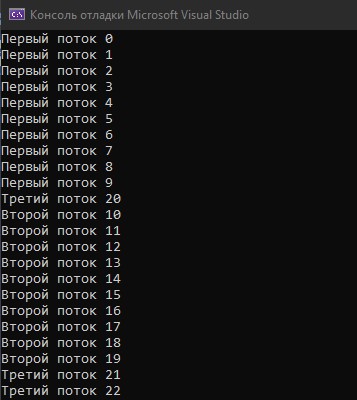


Рисунок 14.1 – Результат работы программы

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.

Листинг программы:

class Program

{

static void Sum()

{

int sum = 0;

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

{

sum += i;

}

Console.WriteLine("Сумма чисел от 1 до 10 равна {0}", sum);

}

static void Main(string[] args)

{

Stopwatch sWatch = new Stopwatch();

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

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

sWatch.Start();

thread1.Start();

sWatch.Stop();

Console.WriteLine("Милисекунд затрачено на выполнение 1-ого потока- " + sWatch.ElapsedMilliseconds.ToString());

sWatch.Restart();

thread2.Start();

sWatch.Stop();

Console.WriteLine("Милисекунд затрачено на выполнение 2-ого потока- " + sWatch.ElapsedMilliseconds.ToString());

Console.ReadLine();}}

Таблица 14.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Сумма чисел от 1 до 10 равна 55  Сумма чисел от 1 до 10 равна 55 | Милисекунд затрачено на выполнение 1-ого потока- 6  Милисекунд затрачено на выполнение 2-ого потока- 53 |

Анализ результатов:

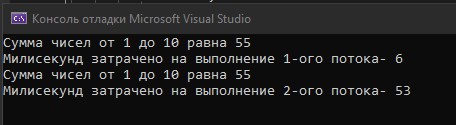


Рисунок 14.2 – Результат работы программы

Источник: собственная разработка