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:

using System;

using System.Threading;

namespace ConsoleApp

{

class Program

{

static void Main(string[] args)

{

Thread thread1 = new Thread(CountNumbers);

Thread thread2 = new Thread(CountNumbers);

Thread thread3 = new Thread(CountNumbers);

thread1.Priority = ThreadPriority.Highest;

thread2.Priority = ThreadPriority.Normal;

thread3.Priority = ThreadPriority.Lowest;

thread1.Start(0);

thread2.Start(10);

thread3.Start(20);

thread1.Join();

thread2.Join();

thread3.Join();

Console.WriteLine("Все потоки завершили выполнение.");

}

static void CountNumbers(object start)

{

int num = (int)start;

lock (Console.Out)

{

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

{

Console.WriteLine($"Поток {Thread.CurrentThread.ManagedThreadId}: {num}");

num++;

Thread.Sleep(100);

}

}

}

}

}

Анализ программы:

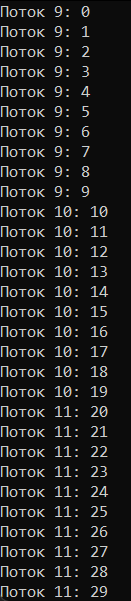


Рисунок 1.1 - Анализ программы

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:

using System;

using System.Threading;

namespace ConsoleApp

{

class Program

{

static void Main(string[] args)

{

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

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

thread1.Start();

thread2.Start();

thread1.Join();

thread2.Join();

Console.ReadLine();

}

static void SumNumbers()

{

int sum = 0;

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

{

sum += i;

}

int threadId = Thread.CurrentThread.ManagedThreadId;

Console.WriteLine($"Поток {threadId}: Sum = {sum}, Затраченное время = {DateTime.Now.Millisecond}мс");

}

}

}

Анализ программы:



Рисунок 1.2 - Анализ 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 ).

Листинг программы:

using System;

using System.Threading;

namespace ConsoleApp

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите значение A:");

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

Console.WriteLine("Введите значение N:");

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

Thread thread1 = new Thread(() => Method1(a, n));

Thread thread2 = new Thread(() => Method1(a, n));

thread1.Start();

thread2.Start();

Method2(a, n);

thread1.Join();

thread2.Join();

}

static void Method1(int a, int n)

{

int sum = 0;

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

{

sum += a + i;

Console.WriteLine($"Поток {Thread.CurrentThread.ManagedThreadId}: Sum = {sum}");

}

}

static void Method2(int a, int n)

{

int product = a;

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

{

product \*= a + i;

Console.WriteLine($"Product = {product}");

}

}

}

}

Анализ программы:

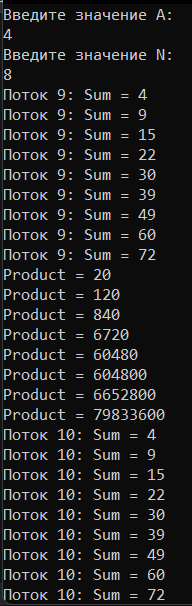


Рисунок 1.3 - Анализ программы

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. Примерный алгоритм реализации: В качестве примера рассмотрим случай, когда в массиве хранятся номера элементов. Основной поток создает нужное число потоков, число которых задается параметром, каждому из которых передается параметр – номер процесса (или номер части массива, обрабатываемой потоком). Функция потока возвращает частичную сумму. Основной поток ожидает завершения всех потоков и находит общую сумму.

Листинг программы:

using System;

using System.Threading;

namespace MultiThreadedSum

{

class Program

{

static void Main(string[] args)

{

int[] arr = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 };

int numThreads = 4;

int[] partialSums = new int[numThreads];

Thread[] threads = new Thread[numThreads];

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

{

int threadNum = i;

threads[i] = new Thread(() => partialSums[threadNum] = SumEvenNumbers(arr, threadNum, numThreads));

threads[i].Start();

}

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

{

threads[i].Join();

}

int totalSum = 0;

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

{

totalSum += partialSums[i];

}

Console.WriteLine("Общая сумма четных чисел: " + totalSum);

}

static int SumEvenNumbers(int[] arr, int startIndex, int step)

{

int sum = 0;

for (int i = startIndex; i < arr.Length; i += step)

{

if (arr[i] % 2 == 0)

{

sum += arr[i];

}

}

return sum;

}

}

}

Анализ программы:



Рисунок 1.4 - Анализ программы

Источник: собственная разработка