**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:

Thread thread1 = new Thread(() => WriteNumFromTo("Первый поток", 1, 9));

Thread thread2 = new Thread(() => WriteNumFromTo("Второй поток", 10, 19));

Thread thread3 = new Thread(() => WriteNumFromTo("Третий поток", 20, 29));

thread1.Priority = ThreadPriority.Highest;

thread2.Priority = ThreadPriority.BelowNormal;

thread3.Priority = ThreadPriority.Lowest;

thread1.Start();

Thread.Sleep(40);

thread2.Start();

thread3.Start();

static void WriteNumFromTo(string mes, int from, int to)

{

for(int i = from; i <= to; i++)

{

Console.WriteLine($"{mes} - {i}");

}

}

Таблица 14.1 – Входные и выходные данные программы задание №1

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

Анализ результатов:

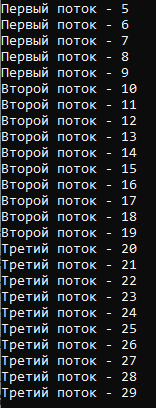


Рисунок 14.1 – Результат выполнения программы задание №1

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.Diagnostics;

int resultThread1 = 0;

Stopwatch sWatch = new Stopwatch();

sWatch.Start();

Thread thread1 = new Thread(() => { resultThread1 = Sum(); });

int resultThread2 = 0;

Thread thread2 = new Thread(() => { resultThread2 = Sum(); });

thread1.Start();

thread2.Start();

thread1.Join();

thread2.Join();

sWatch.Stop();

Console.WriteLine($"Количество затрачиного времени(миллисекунд): {sWatch.ElapsedMilliseconds}");

Console.WriteLine($"Результат выполнения первого потока: {resultThread1}");

Console.WriteLine($"Результат выполнения второго потока: {resultThread2}");

int Sum()

{

int sum = 0;

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

{

sum += i;

}

return sum;

}

Таблица 14.2 – Входные и выходные данные программы задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Количество затрачиного времени(миллисекунд): 10  Результат выполнения первого потока: 55  Результат выполнения второго потока: 55 |

Анализ результатов:

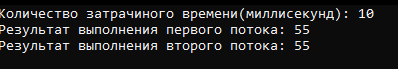


Рисунок 14.2 – Результат выполнения программы задание №2

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+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:

Console.Write("Введите число A: ");

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

Console.Write("Введите N: ");

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

Thread thread1 = new Thread(() =>

{

Console.WriteLine($"Поток 1 - метод 1: {Metod1(numA, n)}");

Console.WriteLine($"Поток 1 - метод 2: {Metod2(numA, n)}");

});

Thread thread2 = new Thread(() =>

{

Console.WriteLine($"Поток 2 - метод 1: {Metod1(numA, n)}");

});

thread1.Start();

thread2.Start();

double Metod1(int num,int n)

{

double result = 0;

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

{

result += Math.Pow(num, i);

}

return result;

}

int Metod2(int num, int n)

{

int result = 1;

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

{

result \*= Convert.ToInt32(Math.Pow(num, i));

}

return result;

}

Таблица 14.3 – Входные и выходные данные программы задание №3

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| A : 12  N: 5 | Поток 2 - метод 1: 271452  Поток 1 - метод 1: 271452  Поток 1 - метод 2: -1073741824 |

Анализ результатов:

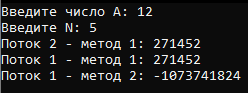


Рисунок 14.3 – Результат выполнения программы задание №3

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:

Console.Write("Введите последнее число последовательности: ");

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

int[] arr = new int[num];

for (int i = 0; i < arr.Length; i++)

{

arr[i] = i + 1;

}

int countThread = GetCountThread(num);

Thread[] threads = new Thread[countThread];

int[] arrResultThreads = new int[countThread];

int startWith = 0;

int sizePart = 5;

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

{

if (sizePart + startWith > arr.Length)

{

sizePart = arr.Length - startWith;

}

int[] partArr = new int[sizePart];

Array.Copy(arr, startWith, partArr, 0, partArr.Length);

startWith += sizePart;

int numThread = i;

threads[i] = new Thread(() =>

{

arrResultThreads[numThread] = SumEvenElemInArr(partArr);

});

threads[i].Start();

}

foreach (Thread thread in threads)

{

thread.Join();

}

Console.WriteLine(arrResultThreads.Sum());

int SumEvenElemInArr(int[] arr)

{

int sum = 0;

for (int i = 0; i < arr.Length; i++)

{

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

{

sum += arr[i];

}

}

return sum;

}

int GetCountThread(int num) => num % 5 != 0 ? Convert.ToInt32(num / 5) + 1 : num / 5;

Таблица 14.4 – Входные и выходные данные программы задание №4

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 8 | 20 |

Анализ результатов:



Рисунок 14.4 – Результат выполнения программы задание №4