**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

Листинг программы:

using System;

using System.Threading;

class Program

{

static AutoResetEvent autoEvent = new AutoResetEvent(false);

static void Main()

{

Thread t1 = new Thread(PrintNumbers);

t1.Priority = ThreadPriority.Lowest;

t1.Start(0);

Thread t2 = new Thread(PrintNumbers);

t2.Priority = ThreadPriority.BelowNormal;

t2.Start(10);

Thread t3 = new Thread(PrintNumbers);

t3.Priority = ThreadPriority.Normal;

t3.Start(20);

Console.ReadLine();

}

static void PrintNumbers(object start)

{

int num = (int)start;

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

{

Console.Write(i + ", ");

Thread.Sleep(1000);

autoEvent.Set();

autoEvent.WaitOne();

}

}

}

Таблица 14.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 10, 0, 20, 21, 11, 1, 22, 12, 2, 23, 13, 3, 24, 14, 4, 15, 25, 5, 16, 26, 6, 17, 7, 27, 18, 8, 28, 9, 19, 29 |

Анализ результатов:

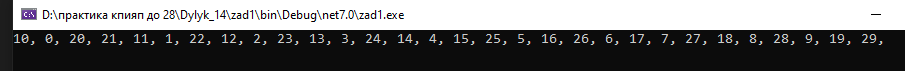


Рисунок 14.1 – Результат работы программы

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:

using System;

using System.Diagnostics;

using System.Threading;

class Program

{

static void Main()

{

Thread t1 = new Thread(SumNumbers);

Thread t2 = new Thread(SumNumbers);

Stopwatch stopwatch = new Stopwatch();

stopwatch.Start();

t1.Start();

t1.Join();

stopwatch.Stop();

Console.WriteLine($"Время выполнения первого потока: {stopwatch.ElapsedMilliseconds} мс");

stopwatch.Reset();

stopwatch.Start();

t2.Start();

t2.Join();

stopwatch.Stop();

Console.WriteLine($"Время выполнения второго потока: {stopwatch.ElapsedMilliseconds} мс");

Console.ReadLine();

}

static void SumNumbers()

{

int sum = 0;

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

{

sum += i;

}

Console.WriteLine($"Сумма чисел от 1 до 10: {sum}");

}

}

Таблица 14.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 3 мс  0 мс |

Анализ результатов:

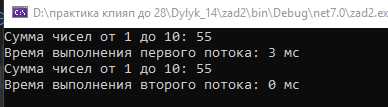


Рисунок 14.2 – Результат работы программы

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:

using System;

using System.Threading;

class Program

{

static Mutex mutex = new Mutex();

static void Main(string[] args)

{

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

double A = Convert.ToDouble(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));

Thread thread3 = new Thread(() => Method2(A, N));

thread1.Start();

thread2.Start();

thread3.Start();

thread1.Join();

thread2.Join();

thread3.Join();

Console.ReadLine();

}

static void Method1(double A, int N)

{

double result = 0;

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

{

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

}

Console.WriteLine("Результат метода 1: " + result);

}

static void Method2(double A, int N)

{

mutex.WaitOne();

double result = 1;

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

{

result \*= Math.Pow(A, i);

}

Console.WriteLine("Результат метода 2: " + result);

mutex.ReleaseMutex();

}

}

Таблица 14.3 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 10  2 | 111  111  1000 |

Анализ результатов:

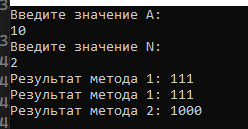
****

Рисунок 14.3 – Результат работы программы

Задание 4. Дана последовательность натуральных чисел {a 0 …a n–1 }. Создать многопоточное приложение для поиска суммы ∑a i , где a i – четные числа.

Листинг программы:

using System;

using System.Linq;

using System.Threading.Tasks;

class Program

{

static void Main(string[] args)

{

Random random = new Random();

int[] numbers = Enumerable.Range(0, 100).Select(\_ => random.Next(0, 10)).ToArray();

int numberOfThreads = Environment.ProcessorCount;

var dividedArray = new int[numberOfThreads][];

int size = numbers.Length / numberOfThreads;

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

{

dividedArray[i] = numbers.Skip(i \* size).Take(size).ToArray();

}

Task<int>[] tasks = new Task<int>[numberOfThreads];

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

{

int[] part = dividedArray[i];

tasks[i] = Task.Factory.StartNew(() => SumEvenNumbers(part));

}

Task.WaitAll(tasks);

int totalSum = tasks.Sum(t => t.Result);

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

}

static int SumEvenNumbers(int[] numbers)

{

return numbers.Where(n => n % 2 == 0).Sum();

}

Таблица 14.4 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 164 |
|  |  |

Анализ результатов:



Рисунок 14.4 – Результат работы программы