1. **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:

namespace Task\_1

{

class Program

{

public static void FirstThread()

{

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

{

Thread.Sleep(120);

Console.Write($"{i} ");

}

}

public static void SecondThread()

{

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

{

Thread.Sleep(120);

Console.Write($"{i} ");

}

}

public static void ThirdThread()

{

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

{

Thread.Sleep(120);

Console.Write($"{i} ");

}

}

public 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;

thread2.Priority = ThreadPriority.AboveNormal;

thread3.Priority = ThreadPriority.Normal;

Console.Write("Первый поток: ");

thread1.Start();

thread1.Join();

Console.Write("\nВторой поток: ");

thread2.Start();

thread2.Join();

Console.Write("\nТретий поток: ");

thread3.Start();

thread3.Join();

Console.WriteLine("\nВыполнено");

}

}

}

Таблица 1.1 – Входные и выходные данные

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

Анализ результатов:

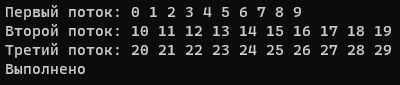


Рисунок 1.1 – Результат работы программы

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:

namespace Task\_2

{

class Program

{

public static int Sum()

{

int sum = 0;

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

{

sum += i;

}

return sum;

}

public static void FirstThread()

{

Console.WriteLine("Вызов метода первым потоком: ");

Console.WriteLine($"Сумма чисел = {Sum()}");

Thread.Sleep(40);

Console.WriteLine("Первый поток выполнен за {0} миллисекунд", DateTime.Now.Millisecond);

}

public static void SecondThread()

{

Console.WriteLine("Вызов метода вторым потоком: ");

Console.WriteLine($"Сумма чисел = {Sum()}");

Thread.Sleep(20);

Console.WriteLine("Второй поток выполнен за {0} миллисекунд", DateTime.Now.Millisecond);

}

public static void Main(string[] args)

{

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

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

thread1.Priority = ThreadPriority.Highest;

thread2.Priority = ThreadPriority.Normal;

thread1.Start();

thread2.Start();

}

}

}

Таблица 2.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1 2 3 4 5 6 7 8 9 10 | 55 924 935 |

Анализ результатов:

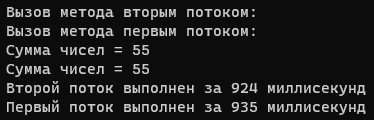


Рисунок 2.1 – Результат работы программы

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:

namespace Task\_3

{

class Program

{

public static void Main(string[] args)

{

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

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

thread1.Start();

thread1.Join();

thread2.Start();

thread2.Join();

Console.WriteLine("Одновременно запущеные потоки");

TwoThread();

}

public static void Sum()

{

Console.Write("Введите число: ");

double a = double.Parse(Console.ReadLine());

Console.Write("Введите количество чисел: ");

int n = int.Parse(Console.ReadLine());

double sum = a;

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

{

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

}

Console.WriteLine($"Сумма = {sum}");

}

public static void Mul()

{

Console.Write("Введите число: ");

double a = double.Parse(Console.ReadLine());

Console.Write("Введите количество чисел: ");

int n = int.Parse(Console.ReadLine());

double mult = a;

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

{

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

}

Console.WriteLine($"Произведение = {mult}");

}

public static void FirstThread()

{

Console.WriteLine("Вызов метода первым потоком: ");

Mul();

}

public static void SecondThread()

{

Console.WriteLine("Вызов метода вторым потоком: ");

Mul();

}

public static void TwoThread()

{

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

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

thread1.Start();

thread2.Start();

}

}

}

Таблица 3.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 4 5  4 6  3 3 4  3 | 1368  5464  123  42 |

Анализ результатов:

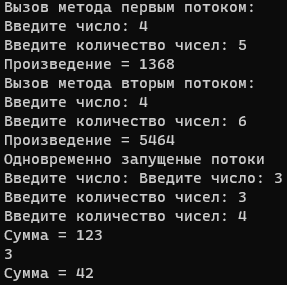


Рисунок 3.1 – Результат работы программы

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.Runtime.CompilerServices;

using System.Text;

namespace Task\_4

{

class Program

{

public static int sum = 0;

public static int n = 0;

public static int a = 0;

public static int b = 0;

public static Barrier barrier = new Barrier(2, (b) =>

{

StringBuilder sb = new StringBuilder();

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

{

sb.Append(" ");

}

});

public static void Main(string[] args)

{

Console.Write("Введите кол-во последовательностей: ");

n = int.Parse(Console.ReadLine());

Console.Write("Введите шаг: ");

b = int.Parse(Console.ReadLine());

Thread myThread = new Thread(new ThreadStart(Count));

myThread.Start();

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

{

a += b;

if (a % 2 == 0)

{

Console.WriteLine("Главный поток:");

Console.WriteLine(sum + a);

barrier.SignalAndWait();

}

}

}

public static void Count()

{

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

{

a += b;

if (a % 2 == 0)

{

Console.WriteLine("Второй поток:");

Console.WriteLine(sum + a);

barrier.SignalAndWait();

}

}

}

}

}

Таблица 4.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 8 3 | 6 12 24 24 36 36 48 48 |

Анализ результатов:

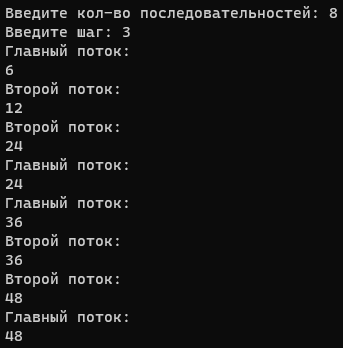


Рисунок 4.1 – Результат работы программы

Источник: собственная разработка