МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии

Департамент цифровых, робототехнических систем и электроники

Отчет к практической работе № 17

Тема: **«Применение класса Thread»**

Дисциплина: **«Алгоритмы и структуры данных»**

|  |  |  |
| --- | --- | --- |
|  |  | **Выполнил:** |
|  |  | Студент группы ПИЖ-б-о-23-1, направление подготовки: 09.03.04 «Программная инженерия»  Панчешный Александр Алексеевич  **Проверил:**  Доцент ДЦРСИЭ  Николаев Евгений Иванович |

Ставрополь 2025

**Цель лабораторной работы:**

Научиться использовать базовые возможности класса потоков Thread.

**Задачи лабораторной работы:**

1. Научиться создавать потоки Thread;
2. Научиться использовать массивы потоков;
3. Научиться осуществлять мониторинг потоков.

**Ход работы**

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Содержимое, созданное искусственным интеллектом, может быть неверным.**Индивидуальное задание:**





**Исходный код:**

using System;

using System.Threading;

class Program

{

*// Делегат для метода вычисления суммы*

delegate void CalculateSumDelegate(int arraySize);

static void Main(string[] args)

{

Console.WriteLine("Программа параллельного вычисления сумм массивов");

const int threadCount = 5; *// Количество потоков в массиве*

const int arraySize = 100000; *// Размер каждого массива*

*// Создаем массив потоков*

Thread[] threads = new Thread[threadCount];

*// Создаем и запускаем потоки*

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

{

int threadNumber = i + 1; *// Номер потока (для вывода информации)*

threads[i] = new Thread(() => CalculateSum(arraySize, threadNumber));

threads[i].Start();

}

*// Мониторинг состояния потоков*

bool allThreadsCompleted = false;

while (!allThreadsCompleted)

{

allThreadsCompleted = true;

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

{

if (threads[i].IsAlive)

{

allThreadsCompleted = false;

break;

}

}

Console.WriteLine("Мониторинг: ожидание завершения потоков...");

Thread.Sleep(500);

}

Console.WriteLine("\nВсе потоки завершили работу!");

Console.WriteLine("Нажмите Enter для выхода...");

Console.ReadLine();

}

*// Метод расчета суммы элементов массива*

static void CalculateSum(int size, int threadNumber)

{

Console.WriteLine($"Поток {threadNumber}: начал вычисления");

*// Генерация случайного массива*

int[] array = GenerateRandomArray(size);

Console.WriteLine($"Поток {threadNumber}: массив сгенерирован");

*// Вычисление суммы*

long sum = 0;

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

{

sum += array[i];

*// Вывод прогресса каждые 10000 элементов*

if (i % 10000 == 0)

{

Console.WriteLine($"Поток {threadNumber}: обработано {i} элементов");

}

}

Console.WriteLine($"Поток {threadNumber}: завершил вычисления. Сумма = {sum}");

}

*// Генерация массива случайных чисел*

static int[] GenerateRandomArray(int size)

{

Random rand = new Random();

int[] array = new int[size];

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

{

array[i] = rand.Next(1, 100); *// числа от 1 до 100*

}

return array;

}

}

**Контрольный вопросы**

**1. В каком пространстве имен определен класс Thread? Поясните назначение класса Thread.**

**Класс Thread** определен в пространстве имен **System.Threading**.

**Назначение класса Thread**:

* Предоставляет возможности для создания и управления потоками выполнения.
* Позволяет выполнять код параллельно в нескольких потоках.
* Дает контроль над приоритетами, состоянием и другими параметрами потоков.

Пример использования:

using System.Threading;

Thread thread = new Thread(SomeMethod);

thread.Start();

**2. Как получить идентификатор текущего потока?**

**Идентификатор потока** можно получить через свойство **ManagedThreadId** класса Thread:

int threadId = Thread.CurrentThread.ManagedThreadId;

Console.WriteLine($"ID текущего потока: {threadId}");

* ManagedThreadId возвращает уникальный числовой идентификатор потока в рамках процесса.
* Это полезно для отладки и логирования параллельных операций.

**3. Какой метод осуществляет запуск метода на выполнение в потоке?**

**Метод**Start() класса Thread:

Thread thread = new Thread(SomeMethod);

thread.Start(); // Запуск потока

* После вызова Start() поток переходит в состояние **Running**.
* Для методов с параметрами используется делегат ParameterizedThreadStart:

Thread thread = new Thread(() => SomeMethod(42));

thread.Start();

**4. Для чего необходимы делегаты при использовании класса Thread?**

**Делегаты** используются для:

1. **Указания метода**, который будет выполняться в потоке.
   * Без делегата поток не знает, какой код запускать.

Пример:

Thread thread = new Thread(new ThreadStart(SomeMethod));

**2. Передачи параметров** в метод потока через ParameterizedThreadStart:

Thread thread = new Thread(new ParameterizedThreadStart(SomeMethodWithParam));

thread.Start(42);

**3. Использования лямбда-выражений** для упрощения синтаксиса:

Thread thread = new Thread(() => SomeMethod(42, "text"));

**5. Как организовать несколько потоков в программе?**

**Способы организации:**

**1. Массив потоков**:

Thread[] threads = new Thread[5];

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

{

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

threads[i].Start();

}

**2. Пул потоков** (через ThreadPool):

ThreadPool.QueueUserWorkItem(state => SomeMethod());

**3. Мониторинг состояния**:

Проверка свойства IsAlive:

if (thread.IsAlive) { /\* поток работает \*/ }

Ожидание завершения через Join():

thread.Join(); // Блокирует текущий поток до завершения

**Важно**:

* Для синхронизации доступа к общим ресурсам используйте lock, Monitor или Mutex.
* В современных приложениях предпочтительнее Task и async/await.

Пример вывода информации о потоках:

Console.WriteLine($"Всего потоков: {Process.GetCurrentProcess().Threads.Count}");