**Выполнили студент 215 группы**

**Марков К.А.**

**Практическое занятие 21**

**Тема: « Создание и запуск задач»**

**Цель работы:** изучить механизм работы с классами задач.

**Задание:**

Вариант 11: Метод вычисления среднего арифметического элементов массива.

class Program

{

static void Main(string[] args)

{

// Создаем массив для демонстрации

double[] numbers = { 1.0, 2.0, 3.0, 4.0, 5.0 };

// Создаем задачу для вычисления среднего арифметического

Task<double> task = Task.Run(() => CalculateAverage(numbers));

// Ожидаем завершения задачи и получаем результат

double average = task.Result;

// Выводим результат на консоль

Console.WriteLine("Среднее арифметическое: " + average);

// Для того чтобы консоль не закрылась сразу после выполнения программы

Console.WriteLine("Нажмите любую клавишу для завершения...");

Console.ReadKey();

}

// Метод для вычисления среднего арифметического элементов массива

static double CalculateAverage(double[] numbers)

{

if (numbers == null || numbers.Length == 0)

{

throw new ArgumentException("Массив не может быть пустым");

}

double sum = 0;

foreach (double num in numbers)

{

sum += num;

}

return sum / numbers.Length;

}

} }

**Вопросы:**

Вопрос 1: Что такое класс Task? Чем класс Task отличается от класса Thread?

Класс Task представляет собой одну операцию, которая не возвращает значение и обычно выполняется асинхронно. `Task` является частью библиотеки TPL (Task Parallel Library), которая облегчает написание многопоточного и асинхронного кода, обеспечивая более высокий уровень абстракции по сравнению с прямым использованием потоков.

Отличия класса Task от класса Thread

- Уровень абстракции: `Task` представляет собой более высокий уровень абстракции, чем `Thread`. `Task` можно рассматривать как действие, которое нужно выполнить, тогда как `Thread` - это конкретный поток исполнения.

- Пул потоков: Задачи, как правило, выполняются на потоках из пула потоков, что уменьшает накладные расходы на создание и уничтожение потоков. Вручную созданные потоки (`Thread`) управляются напрямую программистом, что может быть менее эффективно при большом количестве коротких задач.

- Возвращаемое значение: `Task` может представлять асинхронную операцию с возвращаемым значением через `Task<T>`, тогда как с потоками это управление возвращаемыми значениями требует дополнительного кода.

- Исключения: `Task` облегчает обработку исключений, которые могут быть собраны и выброшены, когда код пытается получить результат задачи или ждет ее завершения.

Вопрос 2: Как получить идентификатор текущей задачи? Как получить идентификатор текущего потока?

Идентификатор текущей задачи:

int taskId = Task.CurrentId.HasValue ? Task.CurrentId.Value : -1;

`Task.CurrentId` возвращает `null`, если код не выполняется внутри `Task`.

Идентификатор текущего потока:

int threadId = Thread.CurrentThread.ManagedThreadId;

Этот метод возвращает идентификатор потока, назначенного управляемой средой CLR.

Вопрос 3: Опишите возможные методы создания и запуска задач.

Задачи можно создать и запустить несколькими способами:

- Использование конструктора `Task` и `Start`

- Использование `Task.Run`

Вопрос 4: Какие параметры задачи может установить программист при реализации задачи? Опишите возможные варианты применения параметра типа TaskCreationOptions.

TaskCreationOptions предоставляет различные опции для управления поведением задач:

- LongRunning: Указывает, что задача, вероятно, будет выполняться длительное время. Это подсказка планировщику для возможного использования дополнительного потока.

ока, чтобы не истощить пул потоков.

- PreferFairness: Призван обеспечить более справедливое распределение времени выполнения между задачами.

- AttachedToParent: Задача присоединяется к родительской задаче и завершается вместе с ней.

- DenyChildAttach: Предотвращает автоматическое присоединение дочерних задач к текущей задаче.

- RunContinuationsAsynchronously: Гарантирует, что продолжение задачи будет запущено асинхронно.

Эти опции позволяют тонко настроить выполнение и поведение задач в зависимости от конкретных требований приложения, управляя распределением ресурсов и обработкой исключений.

**Вывод:** изучил механизм работы с классами задач.