Вопросы 15

1. Что такое TPL? Как и для чего используется тип Task

**Task** **Parallel** **Library** (**TPL**) появился в .NET 4.0. Сейчас это стандарт и основной инструмент для работы с асинхронностью. Любой код использующий более старые подход считается legacy. Основной единицей **TPL** является класс Task из пространства имен System.Threading.Tasks. Класс **Task** в **C#** — это операция, которая выполняется асинхронно и не возвращает значения.

2. Почему эффект от распараллеливания наблюдается на большом количестве элементов?

3. В чем основные достоинства работы с задачами по сравнению с потокми?

Thread это концепция более низкого уровня: если вы напрямую запускаете поток, вы знаете, что это будет отдельный поток, а не выполнение в пуле потоков и т. Д.

Task это больше, чем просто абстракция "где запускать некоторый код", хотя на самом деле это просто "обещание результата в будущем". Итак, несколько разных примеров:

4. Приведите три способа создания и/или запуска Task?

Task.Run(… )

Task t = Task.Factory.StartNew(… )

t.Start()

5. Как и для чего используют методы Wait(), WaitAll() и WaitAny()?

Wait() Ожидает завершения выполнения задачи Task.

WaitAll(Task[]) Ожидает завершения выполнения всех указанных объектов Task.

WaitAny(Task[]) Ожидает завершения выполнения любого из указанных объектов Task.

6. Приведите пример синхронного запуска Task?

Имя\_таска.RunSynchronously();

7. Как создать задачу с возвратом результата?

public Task(Func<TResult> функция)

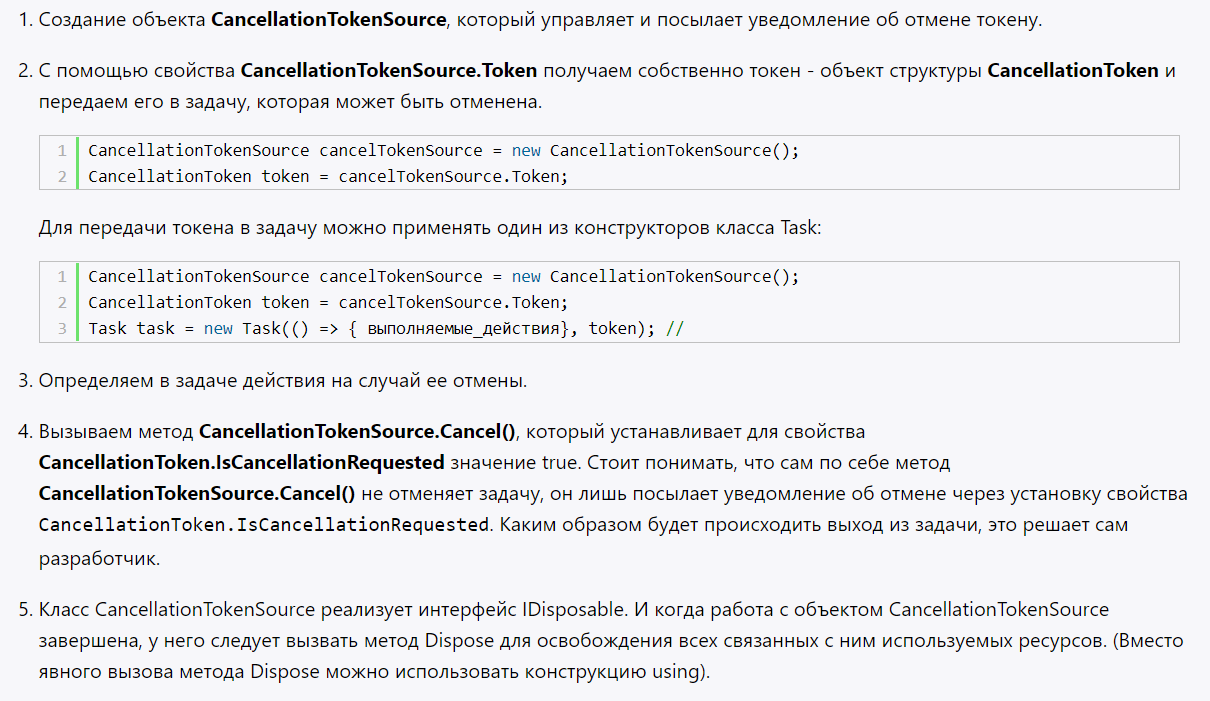
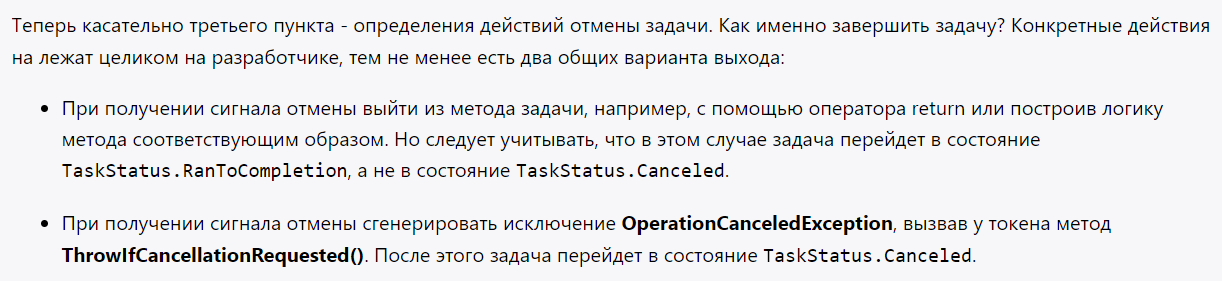
public Task(Func<Object, TResult> функция, Object состояние)

8. Как обработать исключение, если оно произошло при выполнении Task?

Try catch

9. Что такое CancellationToken и как с его помощью отменить выполнение задач?

Параллельное выполнение задач может занимать много времени. И иногда может возникнуть необходимость прервать выполняемую задачу. Для этого платформа .NET предоставляет структуру **CancellationToken** из пространства имен System.Threading.

10.Как организовать задачу продолжения (continuation task) ?

ContinueWith

[TaskFactory.ContinueWhenAll](https://learn.microsoft.com/ru-ru/dotnet/api/system.threading.tasks.taskfactory.continuewhenall)  - чтобы выполнить продолжение после завершения всех предшествуемых задач,

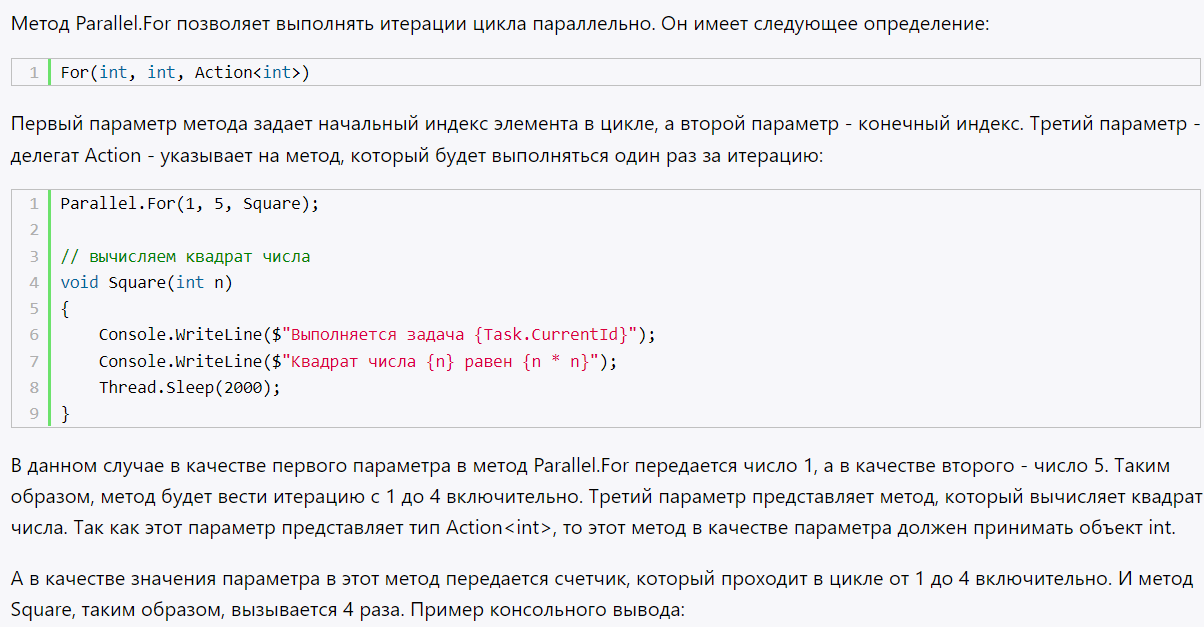
11.Как и для чего используется объект ожидания при создании задач продолжения?

Чтоб какая-нибудь задача не закончила свою работу раньше времени или другая не началась раньше времени

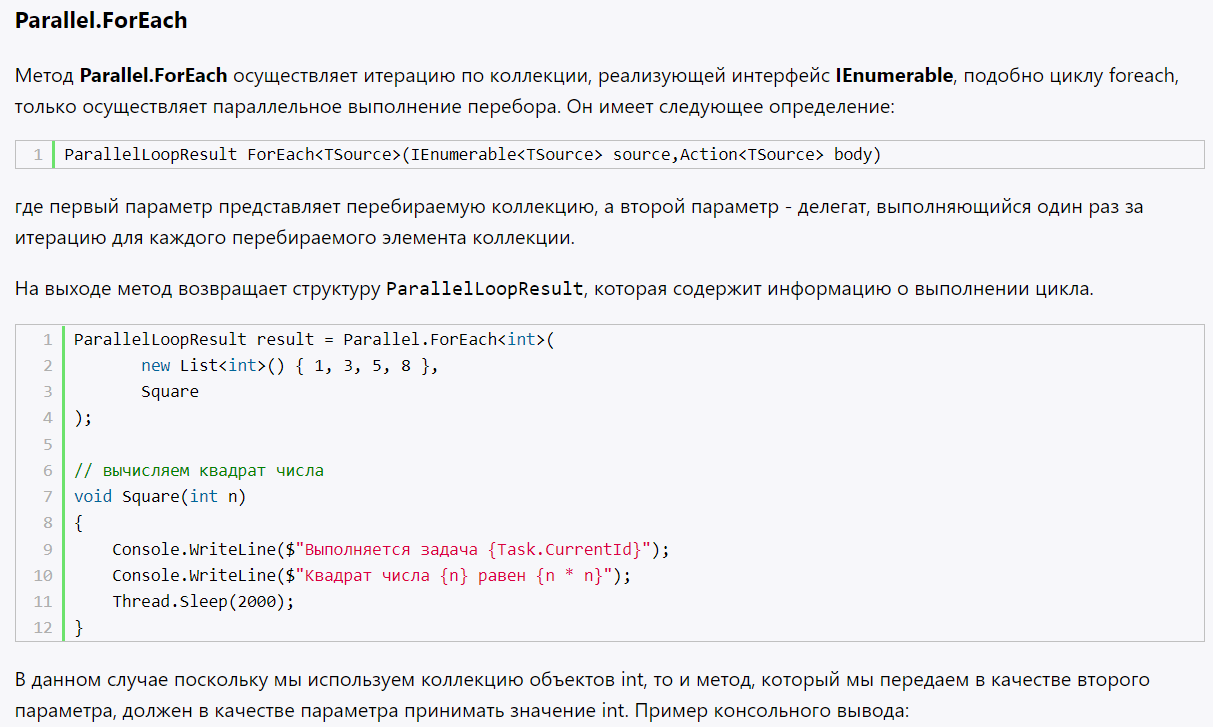
12.Поясните назначение класса System.Threading.Tasks.Parallel?

Предоставляет поддержку параллельных циклов и областей. А также параллельного программирования

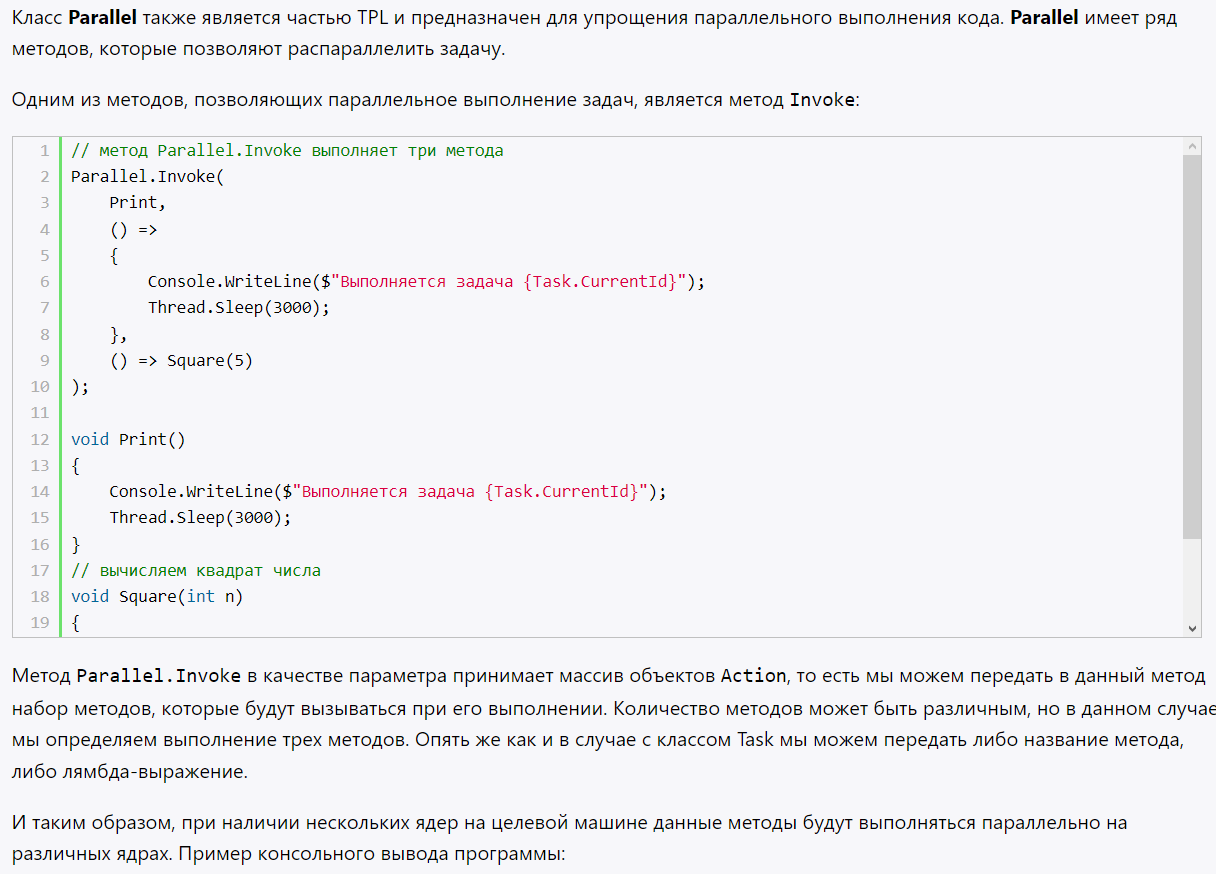
13.Приведите пример задачи с Parallel.For(int, int, Action)

В лабе 

14.Приведите пример задачи с Parallel.ForEach

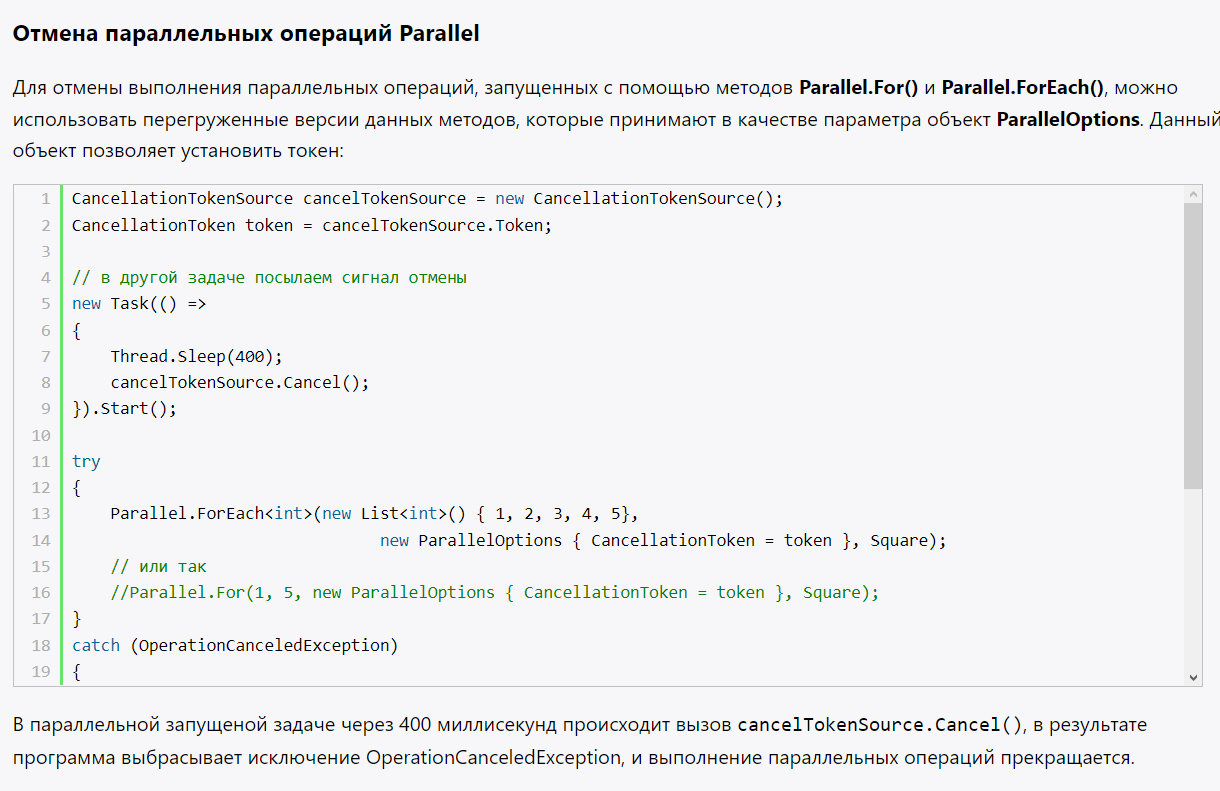


15.Приведите пример с Parallel.Invoke()



16.Как с использованием CancellationToken отменить параллельные операции?

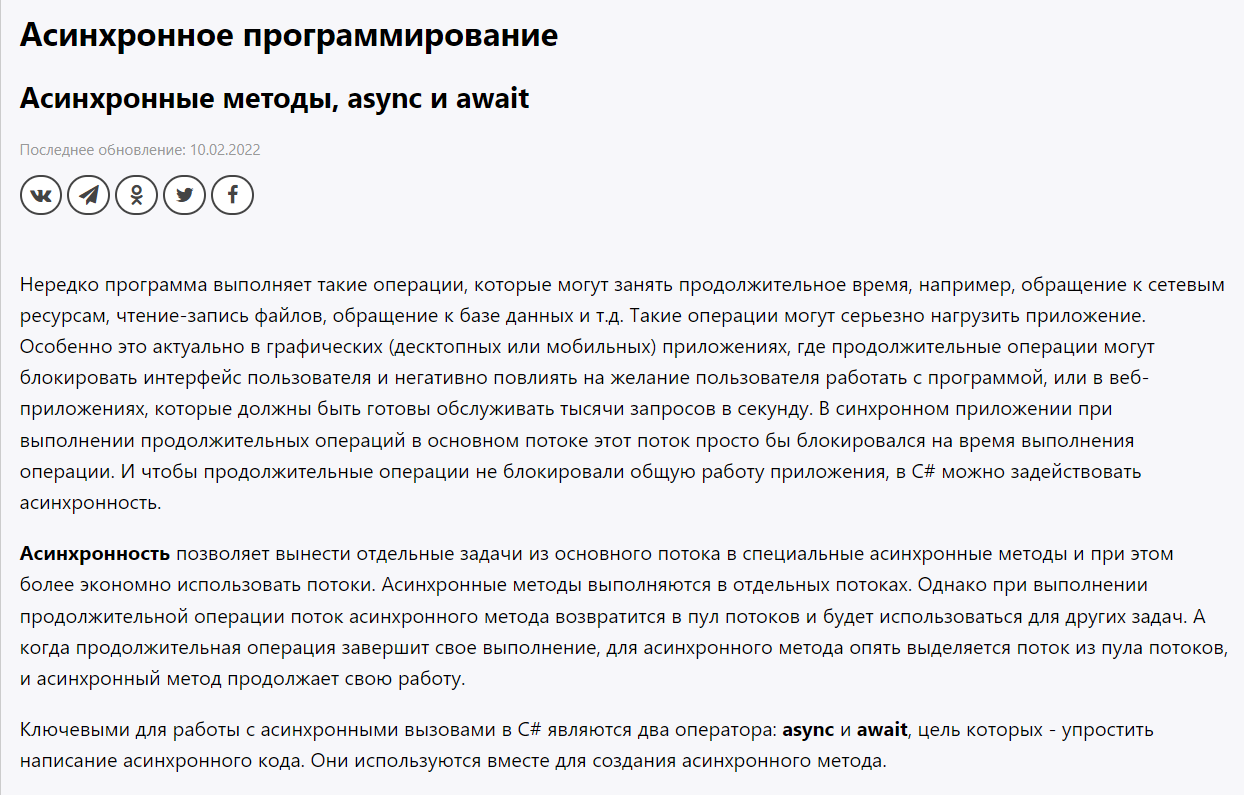
https://metanit.com/sharp/tutorial/12.5.php



17.Для чего используют BlockingCollection, в чем ее особенность?

**BlockingCollection** — **используется** в сценариях, когда одни потоки заполняют коллекцию, а другие извлекают из **нее** элементы. Типичный пример — пополняемая очередь задач. Если в момент запроса очередного элемента коллекция пуста, то читающая сторона переходит в состояние ожидания нового элемента (polling).

18.Как используя async и await организовать асинхронное выполенение метода?



https://metanit.com/sharp/tutorial/13.3.php