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

Tpl – библиотека параллельных задач.

Task –данный класс описывает отдельную задачу, которая запускается асинхронно в одном из потоков пула потоков.

1. **Почему эффект от распараллеливания наблюдается на большом количестве элементов?**

Потому что при распараллеливании мы создаем несколько независимых задач.

1. **В чем основные достоинства работы с задачами по сравнению с потоками?**

Он очень удобен тем что можно перестать мыслить о том, как создать поток, когда его создать, когда запустить, как уничтожить поток. За вас всю эту работу выполняют таски.

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

task = new Task(() => Console.WriteLine("Hell"));

task.Start();

Task task2 = Task.Factory.StartNew(() => Console.WriteLine("Hell 2"));

Task task3 = Task.Run(() => Console.WriteLine("Hell 3"));

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

Для организования ожидания задач. Приостанавливают исполнение вызывающего потока

до тех пор, пока не завершится вызываемая задача

1. **Приведите пример синхронного запуска Task?**

Task<int> task\_3\_1 = new Task<int>(()=>FT\_3(t31));

Task<int> task\_3\_2 = new Task<int>(()=>FT\_3(t32));

Task<int> task\_3\_3 = new Task<int>(()=>FT\_3(t33));

task\_3\_1.Start();

task\_3\_2.Start();

task\_3\_3.Start();

**7. Как создать задачу с возвратом результата?**

Для того чтобы возвратить результат из задачи, достаточно создать эту задачу, используя обобщенную форму Task<TResult> класса Task

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

try

{

Task.WaitAll(Cn, Pr,cn2);

}

catch(Exception ex)

{

Console.WriteLine(ex);

}

finally

{

Cn.Dispose();

Pr.Dispose();

cn2.Dispose();

bc.Dispose();

}

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

[CancellationToken](http://msdn.microsoft.com/en-us/library/system.threading.cancellationtoken.aspx) — маркер отмены; позволяет несколькими способами отслеживать запросы на отмену операции

private static Int32 Sum(CancellationToken ct, Int32 n)

{

Int32 sum = 0;

for(;n > 0; n--)

{

ct.ThrowIfCancellationRequested();

checked { sum += n; }

}

return sum;

}

**10. Как организовать задачу продолжения (continuation task)?**

Task task\_3\_f = task\_3\_3.ContinueWith((Task t) => Console.WriteLine($"Sum (ContinueWith): {task\_3\_1.Result + task\_3\_2.Result + task\_3\_3.Result}"));

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

При выполнении метода могут быть сгенерированы два исключения. Первым из них является исключение ObjectDisposedException. Оно генерируется в том случае, если задача освобождена посредством вызова метода Dispose(). А второе исключение, AggregateException, генерируется в том случае, если задача сама генерирует исключение или же отменяется.

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

14. Приведите пример задачи с Parallel.ForEach (лаба)

15. Приведите пример с Parallel.Invoke()(лаба)

**16. Как с использованием CancellationToken отменить параллельные операции?**

ct.ThrowIfCancellationRequested();

**17. Для чего используют BlockingCollection<T>, в чем ее особенность?**

Поток, добавляющий элементы, блокируется, если коллекция уже заполнена, а поток, удаляющий элементы, блокируется, если она пуста

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

Добавить методу ключевое слово async, перед операцией в методе поставить await

Также стоит отметить, что слово **async**, которое указывается в определении метода, не делает автоматически метод асинхронным. Оно лишь указывает, что данный метод может содержать одно или несколько выражений **await**.