Задачи

Синхронизация

Реализовать решение упрощённой задачи «производитель-потребитель» (буфер не имеет верхней границы) с одним из перечисленных средств синхронизации: атомарные операции, мьютексы, семафоры, мониторы.

- Производители – объекты, кладущие некоторые объекты (например, числа, строки или более сложные объекты-заявки) нестатическими методами в экземпляр класса List<T>;

- Потребители – объекты, извлекающие заявки из экземпляра List<T> в нестатических методах;

- Между двумя последовательными добавлениями у одного и того же производителя или двумя последовательными изъятиями у одного и того же потребителя вставляется пауза (например, с помощью Thread.Sleep);

- Количество производителей и потребителей задаётся константами;

- При запуске программы создаются производители и потребители. Они заканчивают работу по нажатию произвольной клавиши. При этом завершение работы производителей и потребителей должно быть корректно реализовано (Thread.Kill таковым не является).

Пул потоков

Реализовать объект ThreadPool, реализующий паттерн «пул потоков» с поддержкой continuation (наподобие https://docs.microsoft.com/en-us/dotnet/api/system.threading.threadpool?view=netframework-4.8 + https://docs.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskfactory?view=netframework-4.8).

Пул потоков:

- Число потоков задаётся константой в классе пула или параметром конструктора.

- У каждого потока есть два состояния: ожидание задачи, выполнение задачи

- Задача — вычисление некоторого значения, описывается в виде Func<TResult> и инкапсулируется в объектах интерфейса IMyTask<TResult>

- Добавление задачи осуществляется с помощью нестатического метода класса пула Enqueue(IMyTask<TResult> a).

- При добавлении задачи, если в пуле есть ожидающий поток, то он должен приступить к ее исполнению. Иначе задача будет ожидать исполнения, пока не освободится какой-нибудь поток.

- Класс должен быть унаследован от интерфейса IDisposable и корректно освобождать ресурсы при вызове метода Dispose().

- Метод Dispose должен завершить работу потоков. Завершение работы коллаборативное, с использованием CancellationToken — уже запущенные задачи не прерываются, но новые задачи не принимаются на исполнение потоками из пула. Возможны два варианта решения --- дать всем задачам, которые уже попали в очередь, досчитаться, либо выбросить исключение во все ожидающие завершения задачи потоки.

- Предусмотреть конфигурирование пула, чтобы была возможна как реализация стратегии Work Stealing, так и Work Sharing.

IMyTask:

- Свойство IsCompleted возвращает true, если задача выполнена

- Свойство Result возвращает результат выполнения задачи

- В случае, если соответствующая задаче функция завершилась с исключением, этот метод должен завершиться с исключением AggregateException, содержащим внутри себя исключение, вызвавшее проблему

- Если результат еще не вычислен, метод ожидает его и возвращает полученное значение, блокируя вызвавший его поток

- Метод ContinueWith — принимает объект типа Func<TResult, TNewResult>, который может быть применен к результату данной задачи X и возвращает новую задачу Y, принятую к исполнению

- Новая задача будет исполнена не ранее, чем завершится исходная

- В качестве аргумента объекту Func будет передан результат исходной задачи, и все Y должны исполняться на общих основаниях (т.е. должны разделяться между потоками пула)

- Метод ContinueWith может быть вызван несколько раз

- Метод ContinueWith не должен блокировать работу потока, если результат задачи X ещё не вычислен

- ContinueWith должен быть согласован с Shutdown --- принятая как ContinueWith задача должна либо досчитаться, либо бросить исключение ожидающему её потоку.

Ограничения:

- В данной работе запрещено использование TPL, PLINQ и библиотечных классов Task и ThreadPool.

- Все интерфейсные методы должны быть потокобезопасны

- Для каждого базового сценария использования должен быть написан несложный тест (добавление 1 задачи, добавление задач, количественно больших числа потоков, проверка работы ContinueWith для нескольких задач). Для всех тестов обязательна остановка пула потоков.

- Также должен быть написан тест, проверяющий, что в пуле действительно не менее n потоков.