1. [Development and Build Tools](https://elearn.epam.com/courses/course-v1:RD_CIS+BDT_RUS+0122/course/)

Написать программу, которая принимает из командной строки последовательность символов (строку) в качестве аргумента и выводит в консоль максимальное количество неодинаковых последовательных символов в строке

1. [Basic of .NET Framework and C#](https://elearn.epam.com/courses/course-v1:RD_CIS+BNFC_RU+0122/course/)

Написать программу, принимающую из командной строки целое число в десятичной системе, и основание новой системы счисления (от 2 до 20), вывести в консоль преобразованное в эту систему исходное число.

1. Interfaces and abstract classes

https://elearn.epam.com/courses/course-v1:RD\_CIS+IAC\_RUS+0122/courseware/03441c1813f2406483563b188e92a639/76c29947e8034cc084c29a1b0b60e903/

(Flight)

Создать структуру "Координата", определяющую 3D координаты некоторого объекта (положительные числа). Определить интерфейс IFlyable с методами FlyTo(новая точка), GetFlyTime(новая точка). Создать классы "Птица", "Самолет", "Дрон", реализующие данный интерфейс и содержащие как миминум поле "Текущее положение".

Использовать следующие предположения: птица летит все расстояние с постоянной скоростью в диапазоне 0-20 км/ч (заданной случайно), самолет увеличивает скорость на 10 км/ч каждые 10 км полета от начальной скорости 200 км/ч., дрон зависает в воздухе каждые 10 минут полета на 1 минуту. Исходя из задачи, ввести дополнительные ограничения(например, невозможность полета дрона на дальность более чем на 1000 км). Методы и введенные ограничения описать в комментариях.

1. [OOP](https://elearn.epam.com/courses/course-v1:RD_CIS+OOP_RUS+0122/course/)

https://elearn.epam.com/courses/course-v1:RD\_CIS+OOP\_RUS+0122/courseware/014b6ec41cb545f6aaa93ec6e60931f7/12ed964d740041b2a9f815a901ef7a48/2?activate\_block\_id=block-v1%3ARD\_CIS%2BOOP\_RUS%2B0122%2Btype%40vertical%2Bblock%40082edbbce3f14b79aa7fac2af92adfb4

Для создания программы по управлению автопарком нужно реализовать следующие сущности в виде отдельных классов: "Двигатель"(включает в себя поля мощность, объем, тип, серийный номер), "Шасси"(количество колес, номер, допустимая нагрузка), "Трансмиссия" (тип, количество передач, производитель).

С использованием этих классов реализовать сущности "Легковой автомобиль", "Грузовик", "Автобус", "Скутер" (отличающиеся уникальными полями), и обеспечить вывод полной информации об объектах этих типов.

1. Collections

<https://elearn.epam.com/courses/course-v1:RD_CIS+Collections_RU+0122/courseware/14f8880cc5f24d8185ca3da231ea0e89/1617c89d04f848a3894e6f11e689f54a/2?activate_block_id=block-v1%3ARD_CIS%2BCollections_RU%2B0122%2Btype%40vertical%2Bblock%40241ed35379e24d63871aff7bc52cadd7>

## Материалы

1. [Коллекции](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/collections)
2. [LINQ](https://metanit.com/sharp/tutorial/15.1.php)
3. [LINQ in C#](https://docs.microsoft.com/ru-ru/dotnet/csharp/linq/linq-in-csharp)
4. [Xml parser](https://metanit.com/sharp/tutorial/16.2.php)
5. [Json parser](https://www.c-sharpcorner.com/article/working-with-json-string-in-C-Sharp/)
6. Заполнить единую коллекцию, содержащую объекты типа "Грузовик", "Легковой автомобиль", "Автобус", "Скутер" (из предыдущего задания по ООП) с различными значениями полей.
7. Записать в соответствующие XML файлы следующую информацию:

* Полную информацию о всех транспортных средствах, объём двигателя которых больше чем 1.5 литров;
* Тип двигателя, серийный номер и его мощность для всех автобусов и грузовиков;
* Полную информацию о всех транспортных средствах, сгруппированную по типу трансмиссии.

1. ObjectOrientedDesignPrinciples

https://elearn.epam.com/courses/course-v1:RD\_CIS+OODP\_RU+0122/courseware/763ec98fb3c44ac5a0d1a9ee141d5ff1/2a9bb2eb72e343aba536a8b1c2362fd2/2?activate\_block\_id=block-v1%3ARD\_CIS%2BOODP\_RU%2B0122%2Btype%40vertical%2Bblock%406ddc85a722514a16b65438c3152f606c

Напишите программу, принимающую из консоли следующую информацию об автомобиле: марка, модель, количество, стоимость одной единицы. После ввода наименований автомобилей, программа должна запросить у пользователя команду. При получении команд программа должна выдать следующую информацию:

count types - количество марок автомобилей;

count all - общее количество автомобилей;

average price - средняя стоимость автомобиля;

average price type - средняя стоимость автомобилей каждой марки (марка задается пользователем), например average price volvo

При получении команды exit программа должна завершиться. Использовать паттерны проектирвоания Singleton, Command

7.

1. Unit test frameworks

## Материалы

1. [Написание юнит тестов](https://docs.microsoft.com/ru-ru/visualstudio/test/unit-test-basics?view=vs-2019#:~:text=The%20AAA%20(Arrange%2C%20Act%2C,to%20the%20method%20under%20test.) - обратите внимание на стандарт написание тестов ААА
2. [Unit testing best practice](https://docs.microsoft.com/ru-ru/dotnet/core/testing/unit-testing-best-practices) - обратите внимание на характеристики хорошего модульного теста (по английски это будет звучать как FIRST)
3. [NUnit](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-nunit)
4. [XUnit](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-dotnet-test)
5. [MSTest](https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-with-mstest)
6. [Сравнение фреймворков](https://www.jondjones.com/architecture/unit-testing/nunit/ms-test-vs-nunit-vs-xunit-which-ones-the-best-you-decide/)

## Практические задания

Доработать приложение из первой задачи Basic of .NET framework and C#, выделив функциональность подсчета максимального количества неодинаковых последовательных символов в строке в отдельный метод. Расширить функциональность, добавив еще два метода: 1. определяющий максимальное количество последовательных одинаковых букв латинского алфавита в строке 2. определяющий максимальное количество последовательных одинаковых цифр

Для каждого метода написать Unit Tests (использовать фреймворк NUnit, XUnit или MSTest на выбор и следовать AAA и FIRST). При создании тестов особое внимание обратить на классы эквивалетности.

### Пять принципов чистых тестов (F.I.R.S.T. Principles)

**Быстрота (Fast)**. Тесты должны выполняться быстро. Все мы знаем, что разработчики люди, а люди ленивы, поскольку эти выражения являются “транзитивными”, то можно сделать вывод, что люди тоже ленивы. А ленивый человек не захочет запускать тесты при каждом изменении кода, если они будут долго выполняться.

**Независимость (Independent).**Результаты выполнения одного теста не должны быть входными данными для другого. Все тесты должны выполняться в произвольном порядке, поскольку в противном случае при сбое одного теста каскадно “накроется” выполнение целой группы тестов.

**Повторяемость (Repeatable).**Тесты должны давать одинаковые результаты не зависимо от среды выполнения. Результаты не должны зависеть от того, выполняются ли они на вашем локальном компьютере, на компьютере соседа или же на билд-сервере. В противном случае найти концы с концами будет весьма не просто.

**Очевидность (Self-Validating).**Результатом выполнения теста должно быть булево значение. Тест либо прошел, либо не прошел и это должно быть легко понятно любому разработчику.  Не нужно заставлять людей читать логи только для того, чтобы определить прошел тест успешно или нет.

**Своевременность (Timely).**Тесты должны создаваться своевременно. Несвоевременность написания тестов является главной причиной того, что они откладываются на потом, а это “потом” так никогда и не наступает. Даже если вы и не будете писать тесты перед кодом (хотя этот вариант уже доказал свою жизнеспособность) их нужно писать как минимум параллельно с кодом.