# Задача 1. Тетрис

## Updates:

15.09.2015 Файл с примерами обновился, исправлена ошибка в примерах.

## Кратко

Нужно смоделировать игру в тетрис на языке C#. На вход мы подадим размеры стакана (игрового поля), последовательность падающих фигурок и последовательность команд по управлению фигурками. Твоя программа должна эффективно моделировать передвижение фигурок по правилам тетриса и уметь считать заработанные очки.

Но задача была бы слишком стандартной в таком виде! Поэтому вот дополнительные требования — все классы в решении должны быть неизменяемыми. В частности, все поля должны быть помечены модификатором readonly, а сеттеры свойств закрыты. Из сторонних библиотек используй [System.Collections.Immutable](https://www.nuget.org/packages/System.Collections.Immutable/) для моделирования игрового поля и [Newtonsoft.Json](https://www.nuget.org/packages/Newtonsoft.Json/) для чтения входных данных. Разберись самостоятельно с этими библиотеками и концепцией неизменяемости.

Наши приоритеты при оценке:

1. Корректность решения;
2. Скорость работы и чистота кода.

Проверка автоматизирована, поэтому строго соблюдай все требования, описанные ниже. Мы сразу отклоним программу, не соблюдающую требования, даже с дополнительными фичами и красивой графикой (которую мы не требуем).

Бонусом будет использование LINQ там, где это уместно. Это встроенный в C# язык манипулирования данными, познакомиться с которым можно в этом [онлайн практикуме](http://bit.ly/1CL8XT8). Рекомендуем пройти его, даже если ты знаком с LINQ, это займет пару часов и точно будет полезно!

## Подробности

Первый аргумент командной строки программы — это файл с входными данными в формате JSON. Он содержит размеры (Width, Height) стакана, список фигурок (Pieces) и строку команд — по одному символу на каждую команду. Всего есть 6 команд:

A — сдвинуть фигурку на 1 клетку влево.

D — сдвинуть фигурку на 1 клетку вправо.

S — сдвинуть фигурку на 1 клетку вниз.

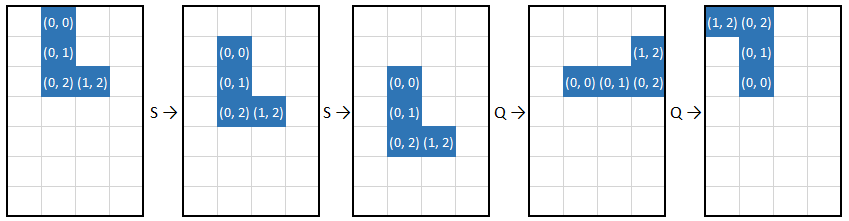
Q — повернуть фигурку против часовой стрелки.

E — повернуть фигурку по часовой стрелке.

P — вывести на консоль текущее состояние стакана.

Ниже более строго описаны правила работы нашего Тетриса.

1. Каждая клетка стакана либо свободна, либо занята. Изначально все свободны.
2. Всегда есть одна текущая фигурка — та, которой мы управляем.
3. Если после применения команды хотя бы одна из ячеек фигурки накладывается на занятую клетку стакана или выходит за границы стакана, то действие команды не выполняется, а вместо этого происходит фиксация фигурки (поле чего команда считается выполненной):
   1. Все ячейки фигурки превращаются в занятые ячейки стакана.
   2. Все полностью заполненные строки стакана исчезают со сдвигом верхних строк вниз. За каждую исчезнувшую строку начисляется 1 очко.
   3. Сверху стакана появляется следующая фигурка. Если это невозможно (одна или несколько её ячеек после появления оказались бы занятыми в стакане), то перед её появлением стакан полностью освобождается. А очки уменьшаются на 10.
4. Фигурка появляется сверху посередине стакана. Точнее, верхняя ячейка фигурки оказывается в самой верхней строке стакана, а расстояние от самой левой ячейки фигурки до левой границы стакана равно или на единицу меньше расстояния от самой правой ячейки фигурки до правой границы стакана.
5. Фигурки появляются циклически в том порядке, в котором они были перечислены во входном файле, начиная с первой.
6. Фигурки во входном файле задаются координатами своих ячеек. Каждая фигурка вращается относительно клетки (0, 0) в этой же системе координат. Центр вращения движется вместе с фигурой.



После фиксации каждой фигурки программа должна вывести номер команды (нумерация с нуля в порядке следования во входном файле), которая привела к фиксации, и текущий счет в одной строке через пробел.

Формат, в котором команда P должна выводить содержимое стакана, демонстрируется примером ниже.

## 

## Ограничения

Ширина и высота стакана не превышают 1 000 000. Количество ячеек в фигуре не превышает 20. Количество фигур не превышает 1 000 000. Количество команд не превышает 100 000 000. Естественно, тесты будут разумными, например, мы не будем заставлять печатать состояние поля размером миллион на миллион.

## Примеры

Пример входного файла:

{

"Width":3, "Height":4,

"Pieces": [

{ "Cells": [ { "X": 0, "Y": 0 }, { "X": 1, "Y": 0 } ] },

{ "Cells": [ { "X": 0, "Y": 0 }, { "X": 0, "Y": 1 } ] } ],

"Commands": "ESSSPQSSSSPAP"

}

На этом файле решение должно вывести на консоль следующее:

3 0

.\*.

.\*.

#..

#..

9 1

\*\*.

...

...

#..

11 -9

.\*.

.\*.

...

...

Решетками отмечены занятые ячейки, точками — свободные, звездочками текущая фигурка.

Больше разнообразных примеров тут: <http://bit.ly/tetris-tests>

## Оформление решения

1. Всё решение должно содержаться в одном cs-файле с кодом на C#, имя которого составлено по шаблону:**<фамилия>\_<имя>.cs** где **фамилия** и **имя** — твои фамилия и имя латиницей.
2. Решение не должно делать потенциально вредоносных действий — обращаться к сети, к файловой системе (естественно, кроме чтения указанного в аргументах входного файла), запускать другие приложения, и т.п.
3. Не должно использовать сторонние библиотеки, кроме указанных в задании выше.
4. Должно в точности соблюдать формат ввода и вывода. Проверь точное совпадение вывода твоего решения с эталонным выводом на предоставленных примерах.

# Задача 2. Описание решения

В файле **<фамилия>\_<имя>.txt** опиши текстом, как устроено решение, на каких исходных данных оно работает особенно эффективно, а на каких ведет себя наихудшим образом. Мы ожидаем 100-150 слов.

# Полезные ссылки

Не уверен, что стоит считать плохим кодом, а что нет? Мы сделали небольшую игру-тренажер, которая научит базовым навыкам приведения кода в порядок. [**http://cleancodegame.github.io**](http://bit.ly/1C9zofM)

А лучше прочитать книгу ["Чистый код" Роберта Мартина](http://bit.ly/1Dwre5P) или хотя бы пробежаться по её [краткому конспекту](http://slidesha.re/1DTGjwg).

Короткий онлайн практикум по языку запросов LINQ: <http://bit.ly/1CL8XT8>

Концепция неизменяемых классов неплохо описана в серии статей тут: <http://weblogs.asp.net/bleroy/immutability-in-c>

Библиотека неизменяемых коллекций: [https://www.nuget.org/packages/System.Collections.Immutable](https://www.nuget.org/packages/System.Collections.Immutable/)

Библиотека для чтения формата JSON: <https://www.nuget.org/packages/Newtonsoft.Json/>

# Вопросы по формулировке

Вопросы по формулировке можно задавать комментариями в этом документе или письмом на почту [kontur-student@kontur.ru](mailto:kontur-student@kontur.ru)