

# Задание

## на программирование правил игры в шахматы

Имеется прототип игры «Шахматы» под iPad с доской и шахматными фигурками, которые можно свободно перемещать по доске.

Задача: запрограммировать правила так, чтобы фигуры можно было перемещать только в соответствии с правилами, включая взятие (поедание) фигур, рокировку, взятие на проходе, определение мата и пата. Для исключения возможных ошибок предусмотреть написание юнит-тестов для правил.

Язык разработки: Objective-C

Среда: Xcode4, iPad simulator 5.0

Для юнит-тестов используется SenTestingKit

### Требования к исполнителю:

- Знание Objective-C
- Понимание ООП
- Наличие XCode4.2
- Четкое представление о том, как реализовать задачу

### Желательно:

- Опыт работы с системами управления версиями. В идеале готовность работать через git посредством bitbucket.org .
- Опыт написания юнит-тестов, TDD приветствуется

Предложения с условиями и аргументированной стоимостью исполнения просьба слать на [job@ri-soft.com](mailto:job@ri-soft.com) .

Подробное ТЗ прилагается.

# Техническое задание

## на программирование правил игры «Шахматы»

### Краткое описание реализации

Имеется ChessGameViewController доски с фигурами, который при попытке хода посылает ChessPass объекту ChessRules (реально сообщение посылается через посредника ChessCoreController), который должен вернуть YES или NO в зависимости от того, разрешен или нет соответствующий ход. Также ChessRules при ходе должен изменить состояние игры ChessState.

Задача программиста:

- 1) Реализовать класс ChessState, в котором должно храниться состояние игры, включая расположение фигур, очередность хода, возможность рокировки и взятия на проходе и пр.
- 2) Реализовать класс ChessRules, который должен проверять соответствие ходов правилам, а также при ходе менять состояние ChessState.
- 3) В ChessCoreController при рокировках и взятии на проходе вызывать у ChessGameViewController методы для перемещения ладьи или для удаления взятой на проходе пешки с доски, или для удаления превратившейся в другую фигуру пешки и добавления одной фигуры (сами методы уже реализованы).

Основной каркас приложения готов, т.е. необходимо дописать классы ChessState, ChessRules и немного ChessCoreController.

Проект написан на Objective-C, использует ARC, создан в XCode4, при разработке предполагается использование iPad Simulator 5.0.

Для написания тестов используется стандартная библиотека SenTestingKit.

## ChessState

1) Должен иметь метод +(id)stateFromNotation:(NSString \*)notation который создаёт состояние используя нотацию Форсайта-Эдвардса:

[http://ru.wikipedia.org/wiki/Нотация\\_Форсайта-Эдвардса](http://ru.wikipedia.org/wiki/Нотация_Форсайта-Эдвардса)

Примеры нотации см. [http://www.thechessdrum.net/PGN\\_Reference.txt](http://www.thechessdrum.net/PGN_Reference.txt) , пункт 16.1.4.

Примечание: поля счетчиков полуходов и номера хода реализовывать не надо, т.е. например начальная расстановка будет иметь вид:

@«rnbqkbnr/pppppppp/8/8/8/PPPPPPPP/RNBQKBNR w KQkq -»

Внутренний формат хранения состояния можно выбрать на своё усмотрение.

Также нужно реализовать метод -(NSString \*)notation , который кодирует внутреннее состояние в строку в соответствии с вышеуказанной нотацией.

2) Интерфейс изменения состояния, который будет использован в ChessRules, реализуется на усмотрение программиста.

3) Для определения очередности хода использовать свойство родительского класса LGState @property (nonatomic) NSUInteger currentPlayer;

4) Предусмотреть свойство для хранения состояния игры (в процессе, пат, мат, победивший игрок).

## ChessRules

При ходе у ChessRules вызывается метод

-(BOOL)makePass:(ChessPass \*)pass withState:(ChessState \*)state;

Т.е. передаётся закодированный ход и текущее состояние игры.

Этот метод (через родительский) вызывает собственный

-(BOOL)isPassLegit:(ChessPass \*)pass withState:(ChessState \*)state

и если тот возвращает YES, то ChessRules должен изменить состояние state в соответствии с ходом.

Т.е. в isPassLegit идёт проверка правил, а в makePass изменение ChessState.

1) Необходимо реализовать все базовые правила игры в шахматы, включая ходы фигур, правила взятия, рокировок, взятия на проходе, запрета на ход под шах, оставления короля под шахом, определения мата и пата, и пр. Правила ничьи на основании 50 ходов без взятия, троекратного повторения позиции и подобные реализовывать не нужно.

## ChessPass (уже написан)

В объекте кодируются:

@property NSUInteger player; (от родительского LGPass) - игрок, который делает ход (0/1).

@property (nonatomic,copy) NSString \*from – поле, с какой ходит фигура, например e2.

@property (nonatomic,copy) NSString \*to – поле, на какую ходит фигура, например e4.

@property (nonatomic,copy) NSString \*promotedFigure – при превращении пешки записывается фигура, в которую она превратилась: (n --- конь, b --- слон, r --- ладья, q --- королева)

На данный момент, пока отсутствует необходимый UI, promotedFigure всегда равен @«q», но на правила это влиять не должно.

## ChessCoreController

Необходимо дописать метод makePass:pass, чтобы при необходимости у ChessGameViewController вызывались методы для удаления/добавления и перемещения фигур.

## Unit tests

Малейшая логическая ошибка в реализации правил может вызывать бурю негодования у игрока и свести на нет все усилия по написанию хорошей игры.

Кроме того, ошибки в правилах могут не проявляться в большинстве партий, т.е. они трудно воспроизводимы и локализуемы. Потому для правил необходимо написание хороших unit-тестов.

Для написания тестов используется стандартная библиотека SenTestingKit. Примеры нескольких тестов включены в проект.