**Rboard**

**Шахматы с графическим интерфейсом**

Содержание

[1. Введение 2](#__RefHeading___Toc312_882511293)

[2. Описание проекта 3](#__RefHeading___Toc314_882511293)

[3. Плюсы для обучения 3](#__RefHeading___Toc316_882511293)

[3.1. Программирование на C# 3](#__RefHeading___Toc318_882511293)

[3.2. Графический интерфейс 3](#__RefHeading___Toc320_882511293)

[3.3. Алгоритмическое мышление 4](#__RefHeading___Toc322_882511293)

[3.4. Автоматизация с Python 4](#__RefHeading___Toc324_882511293)

[4. Цель проекта 4](#__RefHeading___Toc326_882511293)

[5. Перспективы развития 4](#__RefHeading___Toc328_882511293)

[5.1. Игровые режимы 4](#__RefHeading___Toc330_882511293)

[5.2. Улучшение UI 5](#__RefHeading___Toc332_882511293)

[5.3. Расширение логики 5](#__RefHeading___Toc334_882511293)

[5.4. Автоматизация 5](#__RefHeading___Toc336_882511293)

[6. Блок-схема хода коня 5](#__RefHeading___Toc338_882511293)

[7. Пример кода: автоматизация на Python 6](#__RefHeading___Toc340_882511293)

[8. Интеграция C# и Python 6](#__RefHeading___Toc342_882511293)

[9. Заключение 7](#__RefHeading___Toc344_882511293)

[10. Приложения 7](#__RefHeading___Toc346_882511293)

**Автор:** Михайленко Ярослав, 7А класс

### ****1. Введение****

Этот проект представляет собой шахматное приложение, разработанное на языке Python и C# с использованием фреймворка Avalonia для создания кроссплатформенного графического интерфейса. Основные функции приложения включают в себя отображение шахматной доски с фигурами, возможность выбора фигуры и совершения ходов по правилам шахмат, визуальную обратную связь, такую как подсветка ходов и изменение цвета клеток, и обеспечение поочередных ходов игроков (белых и черных). Проект направлен на практическое применение знаний программирования, демонстрацию возможностей C# и Avalonia в разработке игр, и формирование навыков отладки и тестирования (например, проверка корректности ходов коня или пешки). В процессе разработки были освоены следующие концепции: C# и ООП (работа с классами, методами, массивами и обработка событий, использование принципов инкапсуляции, логика ходов вынесена в отдельный класс Logic), работа с графическим интерфейсом (знакомство с фреймворком Avalonia, создание динамического UI, обработка кликов и обновление интерфейса в реальном времени), алгоритмическое мышление (реализация правил шахмат, оптимизация кода для работы с двумерными массивами), и документирование и структурирование кода (четкое разделение логики, интерфейса и инициализации). Этот проект служит основой для дальнейшего развития и добавления новых функций, таких как режим игры против ИИ, сетевая игра, сохранение и загрузка партий, улучшение интерфейса и расширение правил игры.

**Цели проекта:**

* Освоение Python / C# и фреймворка Avalonia.
* Реализация логики шахматных правил.
* Интеграция Python для автоматизации тестирования и анализа данных.
* Создание полноценного приложения с возможностью дальнейшего расширения.

### ****2. Описание проекта****

Приложение представляет собой классические шахматы с графическим интерфейсом. Основные компоненты:

* **Логика игры**: проверка ходов, поочередность игроков, правила для каждой фигуры.
* **Интерфейс**: доска 8×8, фигуры в виде Unicode-символов, интерактивные элементы.
* **Автоматизация**: Python-скрипты для тестирования и генерации отчетов.

**Используемые технологии:**

* Язык C#/Python (для основной логики и UI).
* Фреймворк Avalonia (кроссплатформенный UI).
* Python (для автоматизации рутинных задач).

### ****3. Плюсы для обучения****

#### ****3.1. Программирование на C#****

* Работа с классами, методами, массивами.
* Обработка событий (клики, обновление UI).
* Использование NuGet-пакетов (Avalonia, Material.Avalonia).

#### ****3.2. Графический интерфейс****

* Создание динамической сетки (Grid).
* Настройка стилей (цвета, шрифты, границы).
* Адаптация под разные разрешения экрана.

#### ****3.3. Алгоритмическое мышление****

* Реализация правил шахмат:
  + Ход коня: проверка "буквы Г".
  + Ход пешки: учет направления и взятия.
  + Проверка "шаха" (планируется в будущем).

#### ****3.4. Автоматизация с Python****

* **Тестирование**: Скрипты проверяют корректность ходов, например:

python

* def test\_knight\_move():

assert is\_knight\_move\_valid(0, 0, 2, 1) == True # Верный ход

assert is\_knight\_move\_valid(0, 0, 1, 1) == False # Неверный ход

* **Анализ данных**: Генерация статистики по частоте ошибок в игре.
* **Документирование**: Парсинг кода и создание отчетов в Markdown.

### ****4. Цель проекта****

Проект решает несколько задач:

1. **Учебная**: Закрепление материала по C# и алгоритмам.
2. **Практическая**: Создание работающего приложения "с нуля".
3. **Интеграционная**: Демонстрация взаимодействия C# и Python.

**Пример автоматизации:**  
Для ускорения разработки использовался Python-скрипт, который генерировал часть кода на C# (например, метод GetPieceSymbol).

### ****5. Перспективы развития****

#### ****5.1. Игровые режимы****

* Игра против ИИ (на основе алгоритма Minimax).
* Сетевой режим через WebSocket.

#### ****5.2. Улучшение UI****

* Анимация перемещения фигур.
* Подсказки допустимых ходов.

#### ****5.3. Расширение логики****

* Реализация рокировки и "взятия на проходе".
* Проверка на мат и пат.

#### ****5.4. Автоматизация****

* Добавление CI/CD (GitHub Actions для сборки и тестов).
* Скрипты для визуализации истории партий.

**5.5. Перевод логики на Haskell**

* Пофиксить баги с логикой и добавить мат и пат

### ****6. Блок-схема хода коня****

Алгоритм проверки хода коня (IsKnightMoveValid):

plaintext

┌───────────────────────┐

│ Начало │

└──────────┬────────────┘

▼

┌───────────────────────┐

│ Вычисление разниц: │

│ rowDiff = |fromRow - toRow|

│ colDiff = |fromCol - toCol|

└──────────┬────────────┘

▼

┌───────────────────────┐

│ Проверка условия: │

│ (rowDiff == 2 AND colDiff == 1)

│ OR │

│ (rowDiff == 1 AND colDiff == 2)

└──────────┬────────────┘

├─ Да ──▶ Ход допустим

▼

Ход недопустим

**Пояснение:**  
Конь ходит буквой "Г" — на 2 клетки по одной оси и на 1 по другой.

### ****7. Пример кода: автоматизация на Python****

Скрипт для генерации тестовых данных:

python

import csv

def generate\_test\_cases():

cases = [

{"from": (0, 0), "to": (2, 1), "expected": True}, # Верный ход коня

{"from": (1, 1), "to": (3, 2), "expected": True},

{"from": (0, 0), "to": (1, 1), "expected": False} # Неверный ход

]

with open("knight\_tests.csv", "w") as file:

writer = csv.DictWriter(file, fieldnames=["from", "to", "expected"])

writer.writeheader()

writer.writerows(cases)

Этот файл затем используется в C# для автоматического тестирования.

### ****8. Интеграция C# и Python****

Для вызова Python-скриптов из C# использовался Process.Start:

csharp

using System.Diagnostics;

void RunPythonScript(string scriptPath) {

Process.Start("python", scriptPath);

}

**Применение:**

* Генерация отчетов.
* Пакетное тестирование.

### ****9. Заключение****

Проект показал, что даже в рамках учебной задачи можно:

* Освоить несколько языков (C#, Python).
* Автоматизировать рутинные процессы.
* Создать приложение с потенциалом для масштабирования.

**Рекомендации:**

* Добавить модульные тесты на C# (xUnit).
* Изучить F# для более сложной логики (например, ИИ).

### ****10. Приложения****

1. **Скриншоты интерфейса**.
2. **Примеры Python-скриптов**.
3. **Диаграмма классов** .

**Итог:** Проект стал отличным примером совмещения обучения и практики, а автоматизация на Python ускорила разработку и добавила гибкости.

**Используемые материалы**

При разработке данного шахматного приложения использовались следующие ключевые материалы и технологии, и ниже приведены ссылки на их документацию:

* **C#**: <https://docs.microsoft.com/en-us/dotnet/csharp/> - Документация по языку программирования C# от Microsoft.
* **Avalonia**: [https://avaloniaui.net/docs/](https://www.google.com/search?q=https://avaloniaui.net/docs/) - Официальная документация по фреймворку Avalonia.
* **Git**: <https://git-scm.com/doc> - Официальная документация по системе контроля версий Git.
* **Python**: <https://docs.python.org/3/> - Официальная документация по языку программирования Python.
* **Haskell**: [https://downloads.haskell.org/~ghc/latest/docs/html/index.html](https://www.google.com/search?q=https://downloads.haskell.org/~ghc/latest/docs/html/index.html) - Документация по Haskell.
* **Библиотеки Python для шахмат**:
  + **python-chess**: <https://python-chess.readthedocs.io/en/latest/> - Документация по библиотеке python-chess