**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

Курсовая работа

по дисциплине «Алгоритмы и структуры данных»

Тема Компьютерная логическая игра «Международные шашки»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337. 22/2375-21 ТЗ-01

Листов 57

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2023 г.

**Исполнитель**:

студент гр. ИСТбд-23

*Захарова Виктория Федоровна*

« » 2023 г.

**2023**

**Содержание**

Техническое задание.....................................................................3

Пояснительная записка ................................................................9

Руководство программиста ..........................................................19

Текст программы............................................................................32

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на курсовую работу

по дисциплине «Алгоритмы и структуры данных»

Тема Компьютерная логическая игра «Международные шашки»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337. 22/2375-21 ТЗ-01

Листов 7

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2023 г.

**Исполнитель**:

студент гр. ИСТбд-23

*Захарова Виктория Федоровна*

« » 2023 г.

**2023**

**Введение**

Компьютерная логическая игра «Международные шашки». Игра происходит по средствам взаимодействия игрока с приложением.

**Играют два участника:** компьютер и реальный игрок. Игра ведется против компьютера.

Общая характеристика функциональных возможностей:

1. Регистрация/авторизация пользователей

2. Возможность сыграть партию

3. Сыграть против бота

Приложение должно соответствовать следующим правилам игры.

**Поле и игроки.**

Играют двое игроков. Игра ведется на шашечном поле. Шашечная доска состоит из 100 (10х10) одинаковых клеток, попеременно светлых и тёмных. Шашки расставляются на чёрных полях первых четырёх горизонтальных рядов с каждой стороны. Пользователь ходит белыми шашками и делает первый ход. Далее ходы делаются поочередно.

**Шашки.** Шашки подразделяются на простые и дамки. Дамки маркируются

знаком короны. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Партнерам перед началом игры предоставляется по 20 шашек, одному - белых, другому - черных.

**Стартовая позиция.**

Шашки расставляются на черных полях первых четырех горизонтальных рядов с каждой стороны и должны удовлетворять требованиям.

**Ходы.**

Ходом в партии считается передвижение шашки с одной клетки поля на

другое. Первый ход всегда делает игрок, играющий светлыми. Игроки поочередно делают по одному ходу до тех пор, пока игра не закончится. Шашки разделяются на простые шашки и дамки. Если шашка попадает на клетку, где расположена шашка противника (компьютера), эта шашка считается “съеденной” и убирается с доски. Простая шашка ходит по диагонали вперёд на одну клетку. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Дамка обозначается символом короны. Дамка, в отличие от простой шашки, ходит на любое из свободных (не занятых другими шашками) полей по диагонали в любом направлении (как вперед, так и назад), но становиться может, как и простая шашка, лишь на не занятые другими шашками клетки, причем через свои шашки она перескакивать не может.

**Правила взятия в международных шашках.**

Если простая шашка находится на одной диагонали рядом с шашкой

другого игрока, за которой имеется свободная клетка, она должна быть перенесена через эту шашку на свободную клетку. Шашка другого игрока в этом случае снимается с поля. Если дамка находится на одной диагонали рядом или на расстоянии от шашки другого игрока, за которой имеется одно или несколько свободных клеток, она должна быть перенесена через эту шашку на любую свободную клетку. Шашка другого игрока в этом случае снимается с поля. Взятие шашки другого игрока является обязательным и производится как вперед, так и назад. Взятие считается одним сыгранным ходом. Взятие своих шашек запрещается. Взятие должно быть четко обозначено и проводится в определенном порядке. Оно считается законченным после снятия шашки соперника с поля. Если в процессе взятия шашкой она вновь оказывается на одной диагонали рядом с другой шашкой игрока, за которой имеется свободная клетка, она должна быть перенесена через

вторую, третью и т.д. шашку. Если в процессе взятия дамкой она снова оказывается на одной диагонали рядом или на расстоянии от шашки другого игрока, за которой находится одна или несколько свободных клеток, дамка должна быть обязательно перенесена через вторую, третью и т.д. шашку и занять любую свободную клетку на

той же диагонали за последней взятой шашкой. После завершения взятия взятые шашки другого игрока снимаются с доски в порядке их взятия.

Если есть несколько вариантов взятия, нужно выполнить тот из них, который снимает максимальное количество шашек соперника (независимо от их качества — и простая, и дамка считаются одной шашкой

При взятии применяется правило [турецкого удара](https://ru.wikipedia.org/wiki/%D0%A2%D1%83%D1%80%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%D1%83%D0%B4%D0%B0%D1%80) — если при бое нескольких шашек противника шашка или дамка повторно выходит на уже побитую шашку, то ход останавливается (то есть запрещается дважды брать одну и ту же шашку, при этом можно пересекать дважды одно и то же пустое поле).

**Выигрыш партии.**

Партия считается выигранной в следующих случаях: 1) если у одного из соперников побиты все шашки 2) если шашки одного из участников заперты, и он не может сделать очередной ход.

**1. Основания для разработки**

В качестве оснований для разработки указывается учебный план направления 09.03.02 «Информационные системы и технологии» и распоряжение по факультету.

**2. Требования к программе или программному изделию**

**2.1. Функциональное назначение**

Требуется разработать однопользовательское десктопное приложение по игре в международные шашки с графическим интерфейсом в среде Windows.

Перечень автоматизируемых процессов:

1. Регистрация и авторизация пользователей
2. Отрисовка игрового поля с шашками
3. Обработка событий мыши
4. Проверка правильности ходов
5. Проверка на конец игры

6) Реализация ходов ботом

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

Общая структура приложения должна быть разделена на 2 компонента.

Первый отвечает за авторизацию/регистрацию, за личный кабинет и

переход к игре.

Второй отвечает за саму игру, её отрисовку, проверку на правила и на

конец игры.

2.2.2 Требования к составу функций приложения

В приложении должны быть реализованы в графическом режиме следующие основные функции:

- регистрация/авторизация пользователя;

- шифрование данных;

- отрисовка игрового поля;

- взаимодействие с пользователем;

- интерактивные прием, проверка правильности и отрисовка хода пользователя;

- проверка окончания игры;

- вычисление, проверка правильности и отрисовка хода компьютера;

- информирование пользователя об окончании игры и победителе.

2.2.3 Требования к организации информационного обеспечения, входных и выходных данных

В приложении должен быть реализован графический интерфейс взаимодействия с пользователем. Отдельно выделены папки под графические файлы, заготовку объектов и карты, а также для самого кода. Логин и пароль пользователя должны вводиться с клавиатуры. Логины и пароли зарегистрированных пользователей должны храниться в отдельном файле или базе данных в зашифрованном виде.

**2.3 Требования к надёжности**

Требования к надежности не предъявляются

**2.4 Требования к информационной и программной совместимости**

Рекомендуется к использованию на Windows 10/11.

Используемые инструменты и их версии:

1. Библиотека “tkinter 8.6.”
2. Библиотека “PIL 10.1.0”.
3. Встроенная библиотека “os”
4. Встроенная библиотека “random”
5. Язык программирования python версии 3.9.11
6. Cреда разработки: PyCharm Community Edition 2022.2.3

**2.5 Требования к маркировке и упаковке**

Определяются заданием на курсовую работу.

**2.6 Требования к транспортированию и хранению**

2.6.1 Условия транспортирования

Требования к условиям транспортирования не предъявляются.

2.6 2 Условия хранения

Диск CD-R должен храниться при комнатной температуре, в диапазоне от 20°C до 25°C. Рекомендуется хранить диск в условиях с относительной влажностью воздуха от 20% до 50%. Диск CD-R должен храниться в темном месте, защищенном от прямых солнечных лучей и других источников яркого света. Для предотвращения повреждения диска CD-R рекомендуется хранить его в специальных пластиковых коробках или футлярах, предназначенных для хранения CD-дисков.

2.6 3 Сроки хранения

Срок хранения – до июля 2024 года.

**3. Требования к программной документации**

Определяются заданием на курсовую работу.

**4. Стадии и этапы разработки**

Определяются заданием на курсовую работу.

**5. Порядок контроля и приёмки**

Определяются заданием на курсовую работу.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

Курсовая работа

По дисциплине «Алгоритмы и структуры данных»

Тема Компьютерная логическая игра «Международные шашки»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Пояснительная записка

Р.02069337. 22/2375-21 ПЗ-01

Листов 9

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2023 г.

**Исполнитель**:

студент гр. ИСТбд-23

*Захарова Виктория Федоровна*

« » 2023 г.

**2023**

**Введение**

Наименование приложения: «Международные шашки».

Наименование игры: «Международные шашки».

Условное обозначение приложения: International checkers

Приложение предназначено для игры в Международные шашки

Выбранный подход основывается на рекурсивном поиске лучшего хода с помощью алгоритма минимакс, использовании копирования поля и глобальных переменных для хранения информации о лучшем ходе компьютера. Это позволяет реализовать "интеллектуального" компьютерного противника, который может принимать обоснованные решения на основе оценки возможных ходов.

Реализованное приложение представляет собой программу для игры в международные шашки между игроком и компьютером с графическим интерфейсом, визуализацией доски и ходов.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу

**1.2 Математические методы**

Математическая модель, используемая в реализации функциональности приложения для игры в международные шашки, основана на алгоритме минимакс.

Выбор алгоритма минимакс обосновывается тем, что он позволяет компьютеру оценить все возможные ходы на несколько шагов вперед, просчитывая возможные последующие ходы и их результаты. Он выбирает ход, который будет наиболее выгодным, в зависимости от ситуации.

**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура

В работе выбраны такие структуры данных, как двумерный массив, представляющий игровое поле или состояние доски.  
Кортежи, используемые для хранения ходов в формате (x1, y1, x2, y2), где (x1, y1) и (x2, y2) представляют начальную и конечную позиции фишки.

Основные функции:

Функция регистрации и функция входа;

Функция шифрования и дешифрования;

Функция открытия файла, который хранит данные о пользователях;

Функция начала новой игры и отрисовка поля;

Функция, определяющая чья очередь и координаты шашек, вызывает функцию ход компьютера или ход игрока в зависимости от очереди;

Функция ход компьютера и функция ход игрока выполняет ход и сканирует поле, определяя наличие победителя.

Функция проверка хода вызывается внутри функции ход компьютера и хода игрока, копирует поле, определяет возможные ходы, выбирает наилучшие, добавляет их в список.

Функция ход вызывается внутри функции ход компьютера и функции ход игрока, рисует перемещение шашек, проверяет условие превращения шашки в дамку, если условие выполняется, превращает пешку в дамку, обрабатывает съедание пешки противника.

В качестве модели выбрана матрица размерности 10×10, со следующими правилами заполнения: - «0» – пустая клетка; - «1» – клетка занята черной шашкой; - «2» - клетка занята черной дамкой; - «3» – клетка занята белой шашкой; - «4» - клетка занята белой дамкой.

Матрица представляет текущее состояние доски в игре и используется для отображения и обработки игрового поля, включая ходы, превращения пешек и рубки фигур противника.

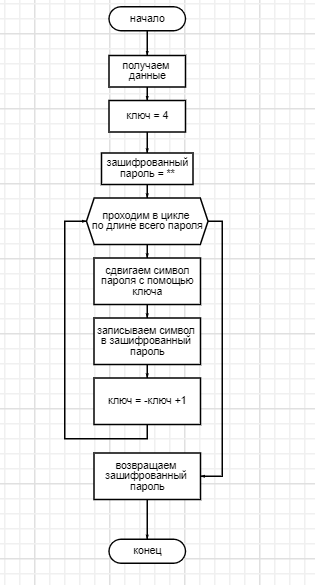
1.3.2 Алгоритм авторизации

Пользователь вводит логин и пароль, алгоритм проверяет соответствуют введенные данные с данными, хранящимися в базе, если нет, то уведомляет о неправильном логине или пароле, если да, то выполняет вход в игру.

1.3.3 Алгоритм регистрации

Данный алгоритм осуществляет регистрацию пользователя, записывая учетные данные, введенные пользователем, в текстовый документ .txt, сообщает об успешной регистрации.

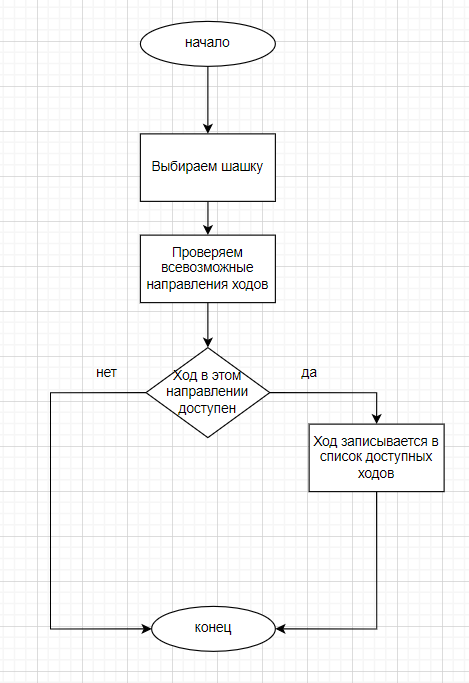
1.3.3 Алгоритм шифрования



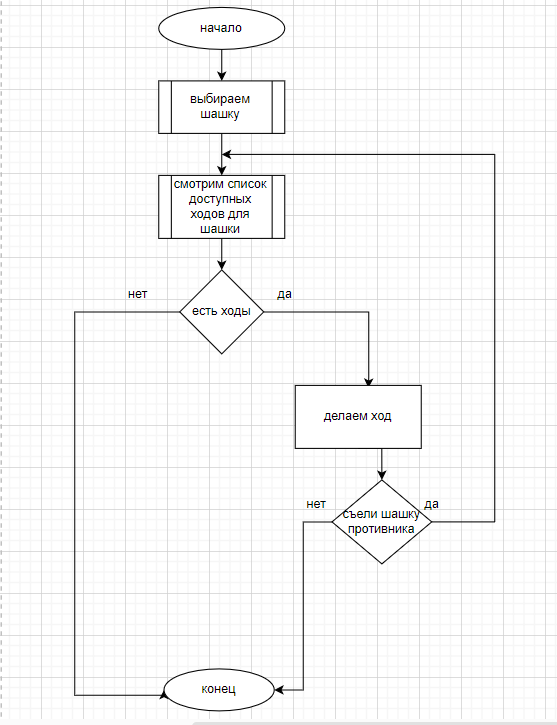
Алгоритм шифрования сдвигает символы пароля на указанное значение 4 и записывает их файл, тем самым защищая персональные данные пользователей.

1.3.4 Алгоритм проверки доступных ходов.

Алгоритм проверяет всевозможные доступные ходы и записывает их в список. В случае их отсутствия игра завершается.



1.3.5 Алгоритм хода игрока



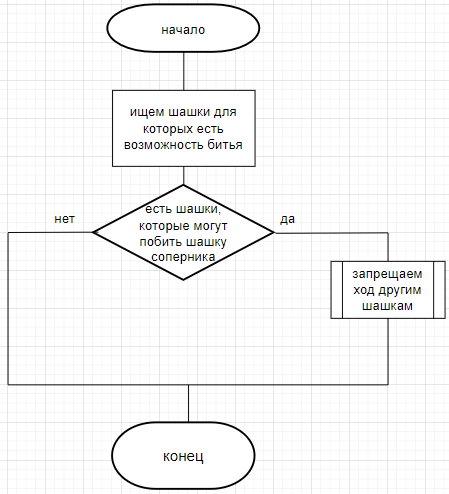
1.3.6 Алгоритм хода компьютера

Компьютер сканирует игровое поле и оценивает все возможные ходы для своих шашек и пешек. Это включает в себя как обязательные ходы (взятие шашек противника), так и обычные ходы. Компьютер использует определенные критерии (количество захваченных шашек противника, количество своих защищенных шашек, близость к превращению в дамку) для оценки каждого возможного хода и выбирает лучший из них. Компьютер выполняет выбранный лучший ход на игровом поле. В случае необходимости (например, если возможны дополнительные взятия шашек противника), алгоритм может повторить оценку возможных ходов и выбор лучшего хода.

1.3.7 Алгоритм превращения пешки в дамку

Алгоритм служит для превращения пешки в дамку. Если пешка достигает последней горизонтали поля ,после завершения хода пешка заменяется на дамку. После превращения пешки в дамку игровое поле обновляется, и новый статус дамки отображается на доске.

1.3.8 Алгоритм определения обязательного битья



1.3.9 Алгоритм конца игры

Алгоритм анализирует текущее состояние игрового поля, представленного матрицей, и проверяет, если все фигуры игрока были срублены или если у игрока больше нет доступных ходов, или если все фигуры компьютера были срублены или если у компьютера больше нет доступных ходов. В зависимости от результата, объявляет о победе одного из игроков, выводит сообщение на экран, завершает игру.

**1.4 Тестирование**

Проводилось тестирование взаимодействия окон авторизации, регистрации, личного кабинета и игры, чтобы переход между ними был корректен. Тестировалось корректность вводимых данных при регистрации и авторизации, проверялась правильность шифрования данных и правильность их записи в файл. Проверялась работоспособность кнопок на окнах и соответствие их выполняемым функциям.

В игре проверялась правильность отрисовки игрового поля с шашками, соответствие хода правилам, чтобы не было возможности сделать запрещённый ход, соответствие выбора шашки (вы играете за белых - можете играть только белыми шашками), актуальность игрового поля (после хода, отрисовка поля должна быть корректной и соответствовать текущему состоянию игрового поля). Функция проверки конца игры должна корректно проверять конец игры и победителя. Передача хода должна быть корректной, т.е. игрок и бот, или игрок и игрок могут ходить только в строгой последовательности, сначала ходят белые, затем чёрные, и ход переходит к другой стороне только тогда, когда текущая сторона выполнила ход. Бот так же должен совершать ходы в рамках правил игры и только шашками своего цвета. После конца игры, партия должна начинаться сначала, с первичной расстановкой фигур на доске. Вся игра должна проходить быстро, без каких-либо зависаний и ошибок.

**2. Источники, использованные при разработке**

1. Wikipedia [Электронный ресурс]: Международные шашки (дата обращения 15.11.2023);
2. YouTube [Электронный ресурс]: Уроки по Tkinter – YouTube – URL:<https://www.youtube.com/playlist?list=PLA0M1Bcd0w8xg_hyqpPpHdbZnPubSyIQ_> (дата обращения: 19.11.2023)
3. Разработка логических компьютерных игр с графическим интерфейсом в среде Питон: учебное пособие для студентов направления 09.03.02 «Информационные системы и технологии» / В.В. Шишкин, Д.С. Афонин. – Ульяновск : УлГТУ, 2023. – 88 с.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

**По дисциплине «Алгоритмы и структуры данных»**

**Тема** Компьютерная логическая игра «Международные шашки»

**Руководство программиста**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337. 22/2375-21 РП-01

Листов 12

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2023 г.

**Исполнитель**:

студент гр. ИСТбд-23

*Захарова Виктория Федоровна*

« » 2023 г.

**2023**

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Компьютерная логическая игра «Международные шашки». Игра происходит по средствам взаимодействия игрока с приложением.

**Играют два участника:** компьютер и реальный игрок. Игра ведется против компьютера.

Общая характеристика функциональных возможностей:

1. Регистрация/авторизация пользователей

2. Возможность сыграть партию

3. Сыграть против бота

Приложение должно соответствовать следующим правилам игры.

**Поле и игроки.**

Играют двое игроков. Игра ведется на шашечном поле. Шашечная доска состоит из 100 (10х10) одинаковых клеток, попеременно светлых и тёмных. Шашки расставляются на чёрных полях первых четырёх горизонтальных рядов с каждой стороны. Пользователь ходит белыми шашками и делает первый ход. Далее ходы делаются поочередно.

**Шашки.** Шашки подразделяются на простые и дамки. Дамки маркируются

знаком короны. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Партнерам перед началом игры предоставляется по 20 шашек, одному - белых, другому - черных.

**Стартовая позиция.**

Шашки расставляются на черных полях первых четырех горизонтальных рядов с каждой стороны и должны удовлетворять требованиям.

**Ходы.**

Ходом в партии считается передвижение шашки с одной клетки поля на

другое. Первый ход всегда делает игрок, играющий светлыми. Игроки поочередно делают по одному ходу до тех пор, пока игра не закончится. Шашки разделяются на простые шашки и дамки. Если шашка попадает на клетку, где расположена шашка противника (компьютера), эта шашка считается “съеденной” и убирается с доски. Простая шашка ходит по диагонали вперёд на одну клетку. При достижении любого поля последней горизонтали, простая шашка превращается в дамку. Дамка обозначается символом короны. Дамка, в отличие от простой шашки, ходит на любое из свободных (не занятых другими шашками) полей по диагонали в любом направлении (как вперед, так и назад), но становиться может, как и простая шашка, лишь на не занятые другими шашками клетки, причем через свои шашки она перескакивать не может.

**Правила взятия в международных шашках.**

Если простая шашка находится на одной диагонали рядом с шашкой

другого игрока, за которой имеется свободная клетка, она должна быть перенесена через эту шашку на свободную клетку. Шашка другого игрока в этом случае снимается с поля. Если дамка находится на одной диагонали рядом или на расстоянии от шашки другого игрока, за которой имеется одно или несколько свободных клеток, она должна быть перенесена через эту шашку на любую свободную клетку. Шашка другого игрока в этом случае снимается с поля. Взятие шашки другого игрока является обязательным и производится как вперед, так и назад. Взятие считается одним сыгранным ходом. Взятие своих шашек запрещается. Взятие должно быть четко обозначено и проводится в определенном порядке. Оно считается законченным после снятия шашки соперника с поля. Если в процессе взятия шашкой она вновь оказывается на одной диагонали рядом с другой шашкой игрока, за которой имеется свободная клетка, она должна быть перенесена через

вторую, третью и т.д. шашку. Если в процессе взятия дамкой она снова оказывается на одной диагонали рядом или на расстоянии от шашки другого игрока, за которой находится одна или несколько свободных клеток, дамка должна быть обязательно перенесена через вторую, третью и т.д. шашку и занять любую свободную клетку на

той же диагонали за последней взятой шашкой. После завершения взятия взятые шашки другого игрока снимаются с доски в порядке их взятия.

Если есть несколько вариантов взятия, нужно выполнить тот из них, который снимает максимальное количество шашек соперника (независимо от их качества — и простая, и дамка считаются одной шашкой

При взятии применяется правило [турецкого удара](https://ru.wikipedia.org/wiki/%D0%A2%D1%83%D1%80%D0%B5%D1%86%D0%BA%D0%B8%D0%B9_%D1%83%D0%B4%D0%B0%D1%80) — если при бое нескольких шашек противника шашка или дамка повторно выходит на уже побитую шашку, то ход останавливается (то есть запрещается дважды брать одну и ту же шашку, при этом можно пересекать дважды одно и то же пустое поле).

**Выигрыш партии.**

Партия считается выигранной в следующих случаях: 1) если у одного из соперников побиты все шашки 2) если шашки одного из участников заперты, и он не может сделать очередной ход.

**1.2 Условия, необходимые для использования приложения**

Рекомендуется к использованию на Windows 10.

При создании программы использовались встроенные библиотеки “os” и “random”, сторонние библиотека “tkinter 8.6.”, “PIL”.

Разработка велась в “PyCharm community edition 2022.3” на версии языка программирования Python 3.9.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Значимых строк кода 651

Структура данных одна – массив.

Использованные библиотеки:

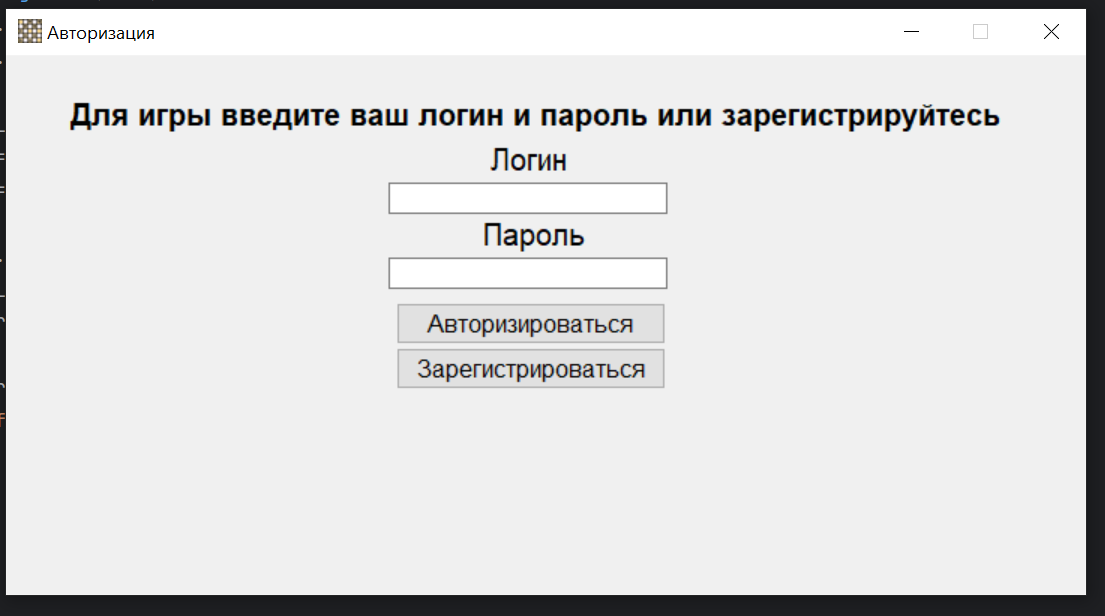
“os” – библиотека функций для работы с операционной системой. Методы, включенные в неё, позволяют определять тип операционной системы, получать доступ к переменным окружения, управлять директориями и файлами

“random” - модуль random позволяет генерировать случайные числа.

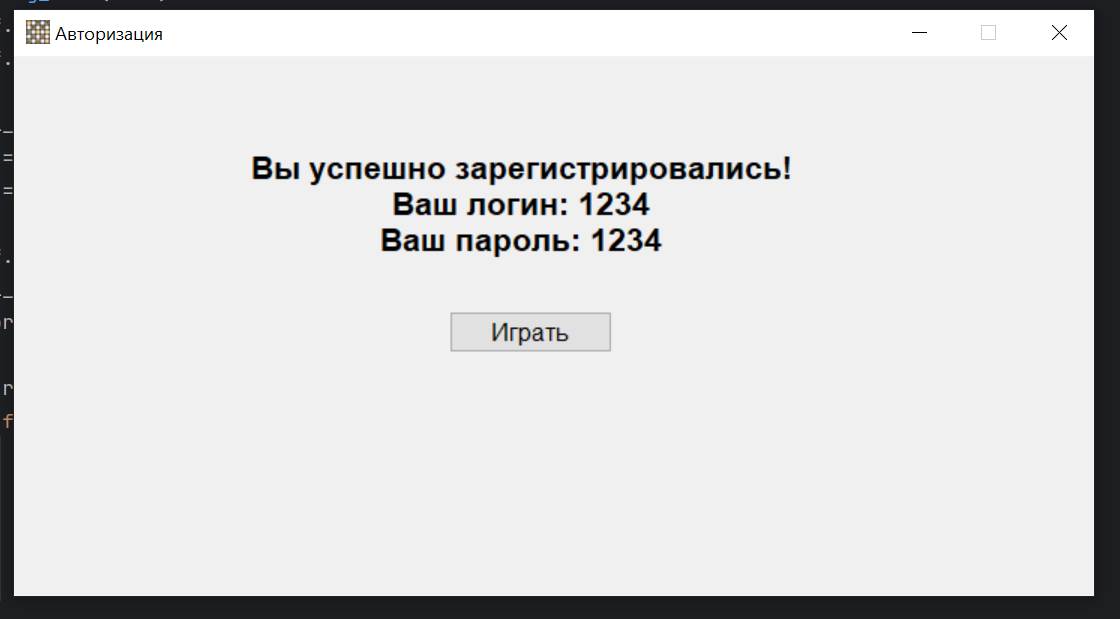
“tkinter 3.6” - библиотека для разработки графического интерфейса на языке Python. Методы, включенные в неё, позволяют создавать окна, размещать на них виджеты, настраивать параметры окна и виджетов.

“PIL” - библиотека языка Python, предназначенная для работы с растровой графикой.

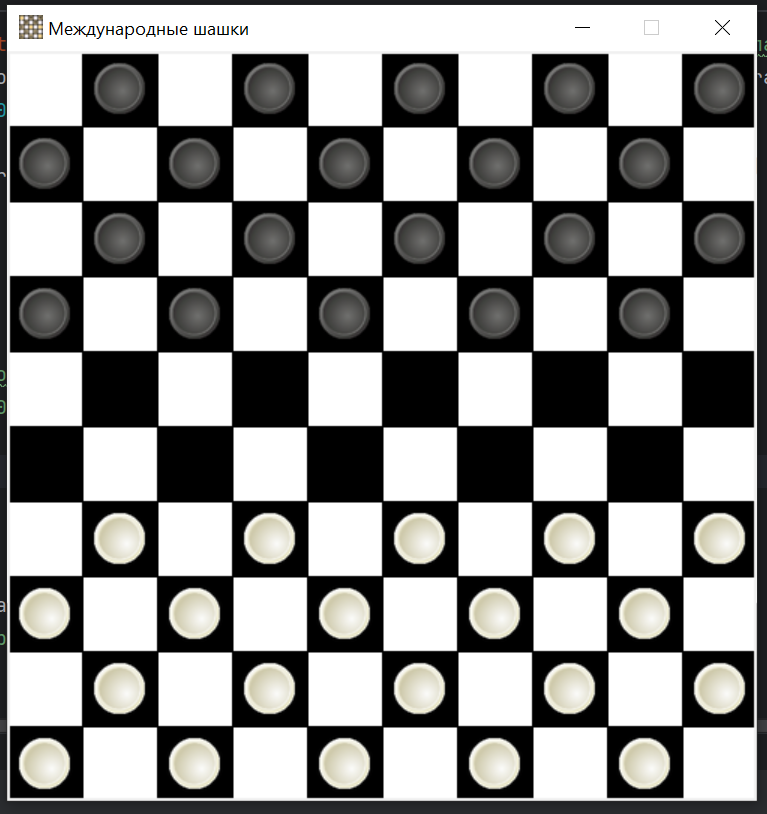
Работа приложения начинается с окна авторизации и регистрации



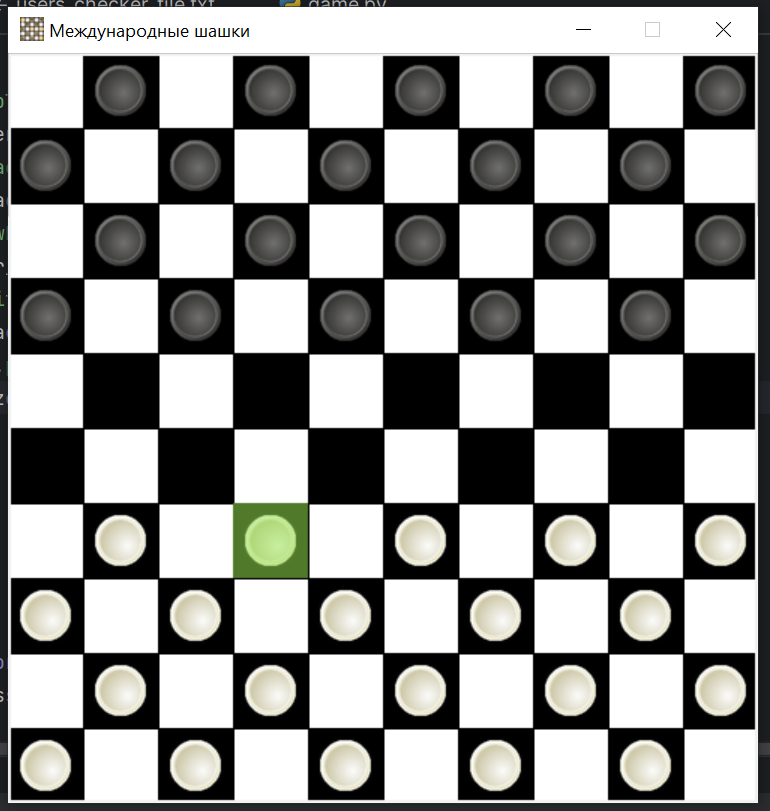
Успешная регистрация



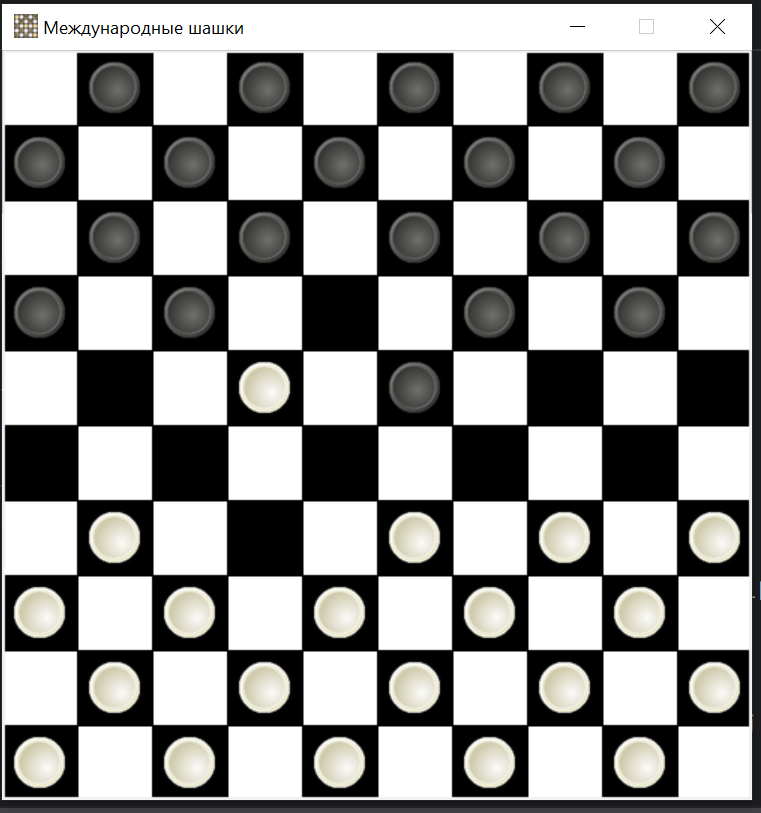
Отрисовка поля

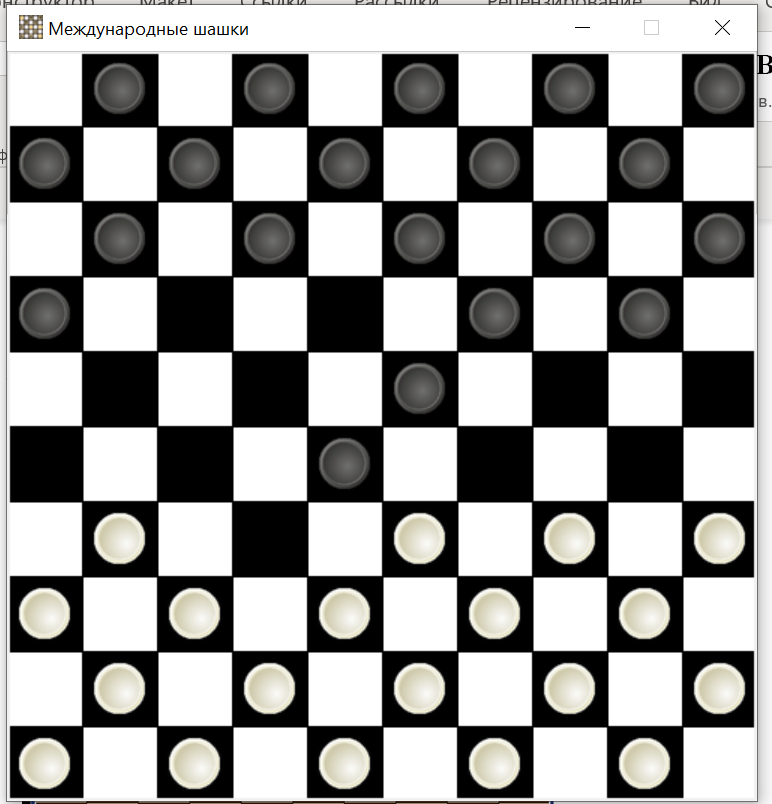


Выбор шашки для хода

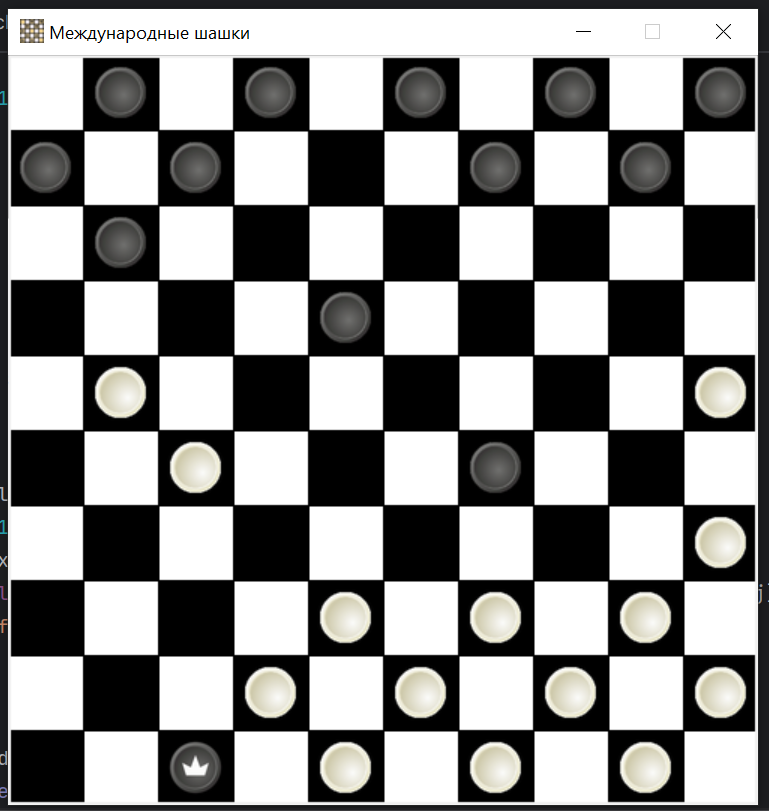


Взятие шашки компьютером

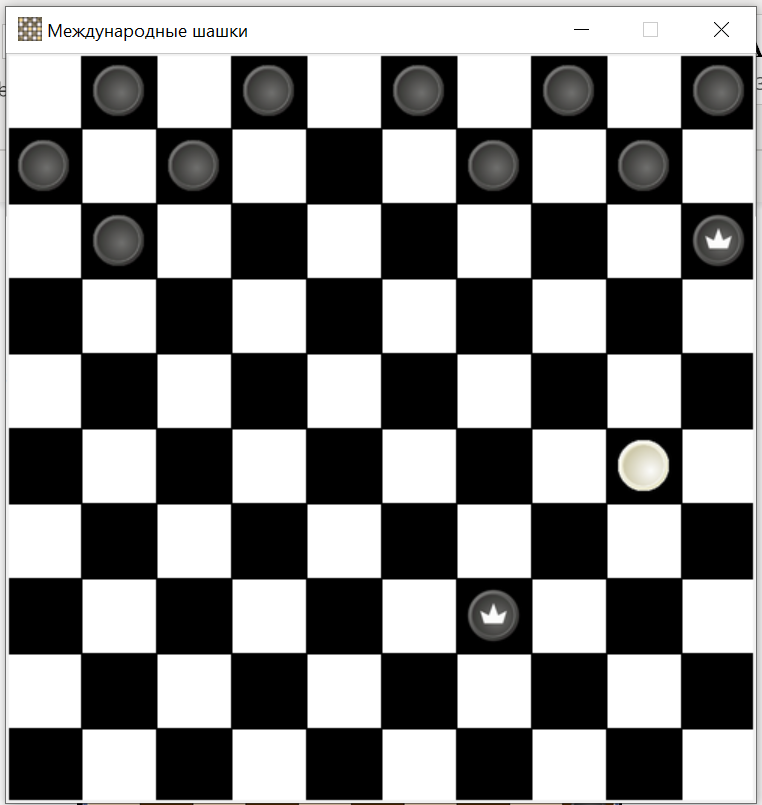




Превращение шашки в дамку



Окончание игры



**2.2 Особенности реализации приложения**

В работе выбраны такие структуры данных, как двумерный массив, представляющий игровое поле или состояние доски.  
Кортежи, используемые для хранения ходов в формате (x1, y1, x2, y2), где (x1, y1) и (x2, y2) представляют начальную и конечную позиции фишки.  
**3. Обращение к программе**

Основные функции:

Функция регистрации и функция входа;

Функция шифрования и дешифрования;

Функция открытия файла, который хранит данные о пользователях;

Функция начала новой игры и отрисовка поля;

Функция, определяющая чья очередь и координаты шашек, вызывает функцию ход компьютера или ход игрока в зависимости от очереди;

Функция ход компьютера и функция ход игрока выполняет ход и сканирует поле, определяя наличие победителя.

Функция проверка хода вызывается внутри функции ход компьютера и хода игрока, копирует поле, определяет возможные ходы, выбирает наилучшие, добавляет их в список.

Функция ход вызывается внутри функции ход компьютера и функции ход игрока, рисует перемещение шашек, проверяет условие превращения шашки в дамку, если условие выполняется, превращает пешку в дамку, обрабатывает съедание пешки противника.

Класс Checkers содержит в себе следующие методы:

1) Метод authorization

Авторизация пользователя, проверяет наличие логина и соответствующего ему пароля в файле с записанными данными пользователей

2) Метод registration

Регистрация пользователя, проверяет наличие логина в файле с записанными данными пользователей и если пользователь с таким логином ещё не зарегистрирован, то вносит данные о пользователе.

3) Метод drawing\_deck

Отрисовка доски, отрисовка шашек в соответствии с данными из массива для представления поля

4) Метод making\_move

Отрисовка хода, удаляет все элементы из окна и заново отрисовывает доску и новое расположение шашек на окне.

5) Метод user\_interact

Обработчик событий, получает данные о нажатии пользователя в окне игры, с помощью алгоритмов проверки хода, проверки обязательного взятия, последовательного боя и превращения шашки в дамку реализует действия на окне игры

6) Метод check\_end

Проверяет наличие ходов для каждой стороны

7) Метод end\_game

Выводит на экран данные о победителе, блокирует нажатия по полю игры и позволяет начать новую игру

Алгоритмы, используемые в программе:

1) password\_code

Шифрование пароля пользователя, сдвигает каждый символ пароля с помощью ключа и возвращает зашифрованный пароль

2) open\_file

Открывает файл с данными о регистрации пользователей, если не находит его, то создаёт новый файл

3) dismiss

Получает окно и позволяет перехватить управление

4) queen\_check

Проверяет, находится ли шашка на нужном поле для превращения в дамку, если да, то меняет её представление в модели поля

5) attack\_check

Получает данные о ходе пользователя. Проходит циклами по всей длине модели поля, если находит шашку или дамку, то проверяет наличие у неё возможности забрать шашку или дамку другого цвета. Для обычной шашки проходит циклом по соседним клеткам, для дамки проходит циклом по всем диагоналям. Если находит возможность взятия, то записывает их в массив. Если данные о ходе пользователя совпадают с записанными во время проверки возможности взятия, то позволяет сделать ход. Если не находит возможность взятия, то позволяет сделать ход.

6)move\_check

Получает данные о ходе ползователя. Проверяет на конечных координатах хода наличие пустого поля, если поле не пустое, то не позволяет сделать ход, иначе проверяет наличие на начальных координатах дамки, если на начальных координатах дамка, то проходит циклом по диагонали хода и если находит там шашку или дамку своего цвета, то не позволяет сделать ход, иначе прибавляет к счётчику шашек 1, если после завершения цикла счётчик шашек равен 1, то позволяет сделать ход и меняет представление шашки другого цвета в массиве на пустое поле, если счётчик шашек больше 1, то не позволяет сделать ход, если равен 0, то проверяет с помощью алгоритма обязательного взятия(attack\_check) возможность взятия шашки или дамки, если не находит возможности взятия, то позволяет сделать ход, иначе не позволяет.

Если на начальных координатах не дамка, то проверяет на сколько клеток сделан ход и находится ли он на диагонали, если ход сделан на 2 клетки и по диагонали, то проверяет наличие на среднем значении между координатами начала и конца хода шашки или дамки другого цвета, если находит, то позволяет сделать ход, иначе не позволяет, если ход сделан на одну клетку то проверяет в правильном ли он направлении, если да проверяет с помощью алгоритма обязательного взятия(attack\_check) возможность взятия шашки или дамки, если не находит возможности взятия, то позволяет сделать ход, иначе не позволяет

7)multijump\_check

Получает данные о только что перемещённой шашке или дамке. Проверяет имеется ли возможность у только что перемещённой шашки или дамки продолжать бой. Для обычной шашки проходит циклом по соседним клеткам, для дамки проходит циклом по всем диагоналям. Если находит возможность взятия, то позволяет совершать последовательный бой.

8) click\_event

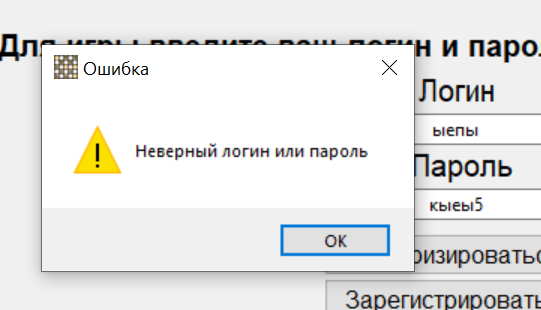
Получает данные о клике пользователя и преобразует их в координаты начала хода. Проверяет очередность хода и находится ли на координатах начала шашка или дамка нужного цвета, если да, то выделяет её и позволяет выбрать координаты конца хода.

9) click\_event\_captured

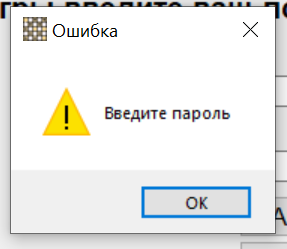
Получает данные о клике пользователя и преобразует их в координаты конца пользователя. Если в этом ходу ещё не было перемещений (переменная self.first\_move) , то позволяет отменить ход нажатием на шашку, иначе не позволяет отменить ход и обязует пользователя продолжать бой. Проверяет с помощью алгоритмов проверки правильности хода (check\_move) корректность хода, если ход корректен, то вызывает метод отрисовки хода(drawing\_move), проверяет с помощью алгоритма проверки последовательного взятия(multijump\_check) возможность последовательного взятия, если не находит, то передаёт ход другой стороне и передаёт управление алгоритму получения данных о начальных координатах (click\_event), иначе меняет переменную первого хода(self.first\_move) и обязует пользователя продолжать взятие. Если обнаруживает, что в этом ходу уже было взятие, то блокирует значение начальных координат и проверяет с помощью алгоритмf проверки правильности хода(move\_check) корректность боя, если находит возможность дальнейшего взятия, то повторяет, иначе передаёт ход другой стороне и передаёт управление алгоритму получения данных о начальных координатах (click\_event).

**4. Сообщения**

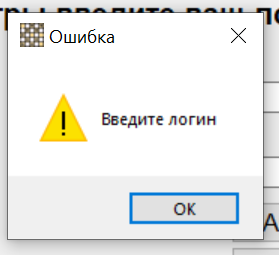
Пользователь не найден или введен неверный пароль



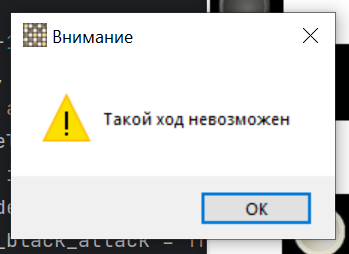
Поле для ввода пароля пустое



Поле для ввода логина пустое



Выбранный игроком ход неверный



**Текст программы**

import os

import random

from tkinter import \*

from tkinter import messagebox

from tkinter import ttk

from PIL import Image, ImageTk

black\_checker\_image = Image.open("black\_checker.png")

black\_checker\_image = black\_checker\_image.resize((40, 40),resample=Image.LANCZOS)

black\_queen\_image = Image.open("black-queen.png")

black\_queen\_image = black\_queen\_image.resize((40, 40),resample=Image.LANCZOS)

white\_checker\_image = Image.open("white\_checker.png")

white\_checker\_image = white\_checker\_image.resize((40, 40),resample=Image.LANCZOS)

white\_queen\_image = Image.open("white-queen.png")

white\_queen\_image = white\_queen\_image.resize((40, 40),resample=Image.LANCZOS)

capture\_image =Image.open("capture.png")

capture\_image = capture\_image.resize((50, 50),resample=Image.LANCZOS)

def password\_code(password):

key = 4

coded\_password = ""

for i in password:

coded\_password\_temp = chr(ord(i) + key)

coded\_password += coded\_password\_temp

key = -key + 1

return coded\_password

def open\_file():

try:

text = open("users\_checker\_file.txt", "r+")

return text

except FileNotFoundError:

try:

text = open("users\_checker\_file.txt", "w")

text.close()

text = open("users\_checker\_file.txt", "r+")

return text

except FileNotFoundError:

text = open("users\_checker\_file.txt", "r+")

return text

def dismiss(win):

win.grab\_release()

win.destroy()

class Checkers:

def \_\_init\_\_(self, main):

self.deck = None

self.first\_click = True

self.main = main

self.users = {}

button\_style = ttk.Style()

button\_style.configure("my.TButton", font="Arial 12")

self.label = Label(text="Для игры введите ваш логин и пароль или зарегистрируйтесь", font="Arial 15 bold")

self.label\_login = Label(text="Логин", font="Arial 15")

self.label\_password = Label(text="Пароль", font="Arial 15")

self.entry\_login = ttk.Entry(width=30, justify="center")

self.entry\_password = ttk.Entry(width=30, justify="center")

self.button\_auth = ttk.Button(text="Авторизироваться", style="my.TButton", command=lambda: self.authorization())

self.button\_reg = ttk.Button(text="Зарегистрироваться", style="my.TButton", command=lambda: self.registrate())

self.button\_back = ttk.Button(self.main, text="Выход", style="my.TButton", command=lambda: self.authorization())

self.label.place(x=40, y=25)

self.label\_login.place(x=320, y=55)

self.entry\_login.place(x=255, y=85)

self.label\_password.place(x=315, y=105)

self.entry\_password.place(x=255, y=135)

self.button\_auth.place(x=260, y=165, width=180)

self.button\_reg.place(x=260, y=195, width=180)

self.x1 = 0

self.y1 = 0

self.x2 = 0

self.y2 = 0

self.deck\_model = [[0, -1, 0, -1, 0, -1, 0, -1, 0, -1],

[-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],

[0, -1, 0, -1, 0, -1, 0, -1, 0, -1],

[-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 1, 0, 1, 0, 1, 0, 1, 0, 1],

[1, 0, 1, 0, 1, 0, 1, 0, 1, 0],

[0, 1, 0, 1, 0, 1, 0, 1, 0, 1],

[1, 0, 1, 0, 1, 0, 1, 0, 1, 0]]

self.white\_checker = ImageTk.PhotoImage(white\_checker\_image)

self.black\_checker = ImageTk.PhotoImage(black\_checker\_image)

self.white\_queen = ImageTk.PhotoImage(white\_queen\_image)

self.black\_queen = ImageTk.PhotoImage(black\_queen\_image)

self.capture\_pic = ImageTk.PhotoImage(capture\_image)

self.white\_move = True

self.black\_move = False

self.first\_move = True

self.move\_without\_attack = False

# Методы для регистрации и авторизации

def authorization(self):

login = self.entry\_login.get()

password\_raw = self.entry\_password.get()

password = password\_code(password\_raw)

if len(login) == 0 and len(password) == 0:

messagebox.showwarning(title="Ошибка", message="Введите логин и пароль")

elif len(login) == 0 and len(password) != 0:

messagebox.showwarning(title="Ошибка", message="Введите логин")

elif len(login) != 0 and len(password) == 0:

messagebox.showwarning(title="Ошибка", message="Введите пароль")

else:

file = open\_file()

a = file.readline()[:-1].split(" ")

while True:

if a != [""]:

self.users[a[0]] = a[1]

a = file.readline()[:-1].split(" ")

else:

break

flag\_reg = False

for i in self.users.items():

login\_check, password\_check = i

if login == login\_check and password == password\_check:

flag\_reg = True

break

if flag\_reg:

for widget in self.main.winfo\_children():

widget.destroy()

Label(self.main, text="Вы успешно авторизировались!", font="Arial 16 bold").place(x=175, y=60)

button = ttk.Button(self.main, text="Играть", style="my.TButton", command=self.drawing\_deck)

button.place(x=290, y=170)

else:

messagebox.showwarning(title="Ошибка", message="Неверный логин или пароль")

def registrate(self):

login = self.entry\_login.get()

password\_raw = self.entry\_password.get()

password = password\_code(password\_raw)

if len(login) == 0 and len(password) == 0:

messagebox.showwarning(title="Ошибка", message="Введите желаемые логин и пароль")

elif len(login) == 0 and len(password) != 0:

messagebox.showwarning(title="Ошибка", message="Введите логин")

elif len(login) != 0 and len(password) == 0:

messagebox.showwarning(title="Ошибка", message="Введите пароль")

else:

file = open\_file()

temp = file.readline()[:-1].split(' ')

while True:

if temp != [""]:

self.users[temp[0]] = temp[1]

temp = file.readline()[:-1].split(' ')

else:

break

flag\_reg = False

for i in self.users.items():

l, p = i

if login == l:

flag\_reg = True

if not flag\_reg:

file = open\_file()

file.seek(0, os.SEEK\_END)

file.write(f'{login} {password}\n')

file.close()

for widget in self.main.winfo\_children():

widget.destroy()

Label(self.main, text=f"Вы успешно зарегистрировались!\nВаш логин: {login}\nВаш пароль: {password\_raw}", font="Arial 16 bold").place(x=155, y=60)

button = ttk.Button(self.main, text="Играть", style="my.TButton", command=self.drawing\_deck)

button.place(x=290, y=170)

else:

messagebox.showwarning(title="Ошибка", message="Такой аккаунт уже существует")

# Метод для вывода доски

def drawing\_deck(self):

self.main.title("Международные шашки")

self.main.geometry("500x500")

cell\_sz = 50

row = 10

col = 10

self.deck = Canvas(self.main, width=cell\_sz \* col, height=cell\_sz \* row)

cell\_colors = ["white", "black"]

color\_index = 0

for rows in range(row):

for cols in range(col):

x1, y1 = cols \* cell\_sz, rows \* cell\_sz

x2, y2 = cols \* cell\_sz + cell\_sz, rows \* cell\_sz + cell\_sz

self.deck.create\_rectangle(x1, y1, x2, y2, fill=cell\_colors[color\_index])

color\_index = not color\_index

color\_index = not color\_index

for rows in range(row):

for cols in range(col):

x1, y1 = cols \* cell\_sz, rows \* cell\_sz

cell = self.deck\_model[rows][cols]

if cell < 0:

if cell == -1:

self.deck.create\_image(x1 + 5, y1 + 5, anchor="nw", image=self.black\_checker)

else:

pass

elif cell > 0:

if cell == 1:

self.deck.create\_image(x1 + 5, y1 + 5, anchor="nw", image=self.white\_checker)

else:

pass

self.deck.pack()

self.user\_interact()

# Метод для отрисовки ходов

def making\_move(self):

self.deck.delete("all")

cell\_colors = ["white", "black"]

color\_index = 0

cell\_sz = 50

row = 10

col = 10

for rows in range(row):

for cols in range(col):

x1, y1 = cols \* cell\_sz, rows \* cell\_sz

x2, y2 = cols \* cell\_sz + cell\_sz, rows \* cell\_sz + cell\_sz

self.deck.create\_rectangle(x1, y1, x2, y2, fill=cell\_colors[color\_index])

color\_index = not color\_index

color\_index = not color\_index

for rows in range(row):

for cols in range(col):

x1, y1 = cols \* cell\_sz, rows \* cell\_sz

cell = self.deck\_model[rows][cols]

if cell < 0:

if cell == -1:

self.deck.create\_image(x1 + 5, y1 + 5, anchor="nw", image=self.black\_checker)

elif cell == -2:

self.deck.create\_image(x1 + 5, y1 + 5, anchor="nw", image=self.black\_queen)

elif cell > 0:

if cell == 1:

self.deck.create\_image(x1 + 5, y1 + 5, anchor="nw", image=self.white\_checker)

elif cell == 2:

self.deck.create\_image(x1 + 5, y1 + 5, anchor="nw", image=self.white\_queen)

self.deck.pack()

# Метод для взаимодействия с пользователями

def user\_interact(self):

self.x1 = 0

self.y1 = 0

self.x2 = 0

self.y2 = 0

def queen\_check(x2, y2):

if self.white\_move:

if self.x2 == 0:

self.deck\_model[x2][y2] = 2

else:

if self.x2 == 9:

self.deck\_model[x2][y2] = -2

def attack\_check(x1, y1, x2, y2):

coord\_checker\_user = [x1, y1, x2, y2]

coord\_checker = []

if self.white\_move:

flag\_white\_attack = False

for x in range(10):

for y in range(10):

if self.deck\_model[x][y] == 1:

for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):

if 0 <= (x + i) <= 9 and 0 <= (y + j) <= 9:

if self.deck\_model[x + i][y + j] == -1 or self.deck\_model[x + i][y + j] == -2:

if 0 <= (x + i \* 2) <= 9 and 0 <= (y + j \* 2) <= 9:

if self.deck\_model[x + i + i][y + j + j] == 0:

flag\_white\_attack = True

coord\_checker.append([x, y, x + i + i, y + j + j])

elif self.deck\_model[x][y] == 2:

for d in range(4):

h = 0

if d == 0:

i = 1

j = 1

elif d == 1:

i = 1

j = -1

elif d == 2:

i = -1

j = 1

elif d == 3:

i = -1

j = -1

while 0 <= x + (i \* h) + i <= 9 and 0 <= y + (j \* h) + j <= 9:

if self.deck\_model[x + (i \* h)][y + (j \* h)] == -1 or \

self.deck\_model[x + (i \* h)][y + (j \* h)] == -2:

if self.deck\_model[x + (i \* h) + i][y + (j \* h) + j] == 0:

flag\_white\_attack = True

h += 1

while 0 <= x + (i \* h) <= 9 and 0 <= y + (j \* h) <= 9:

if self.deck\_model[x + (i \* h)][y + (j \* h)] == 0:

coord\_checker.append([x, y, x + (i \* h), y + (j \* h)])

else:

break

h += 1

if not flag\_white\_attack:

return True

else:

if coord\_checker\_user in coord\_checker:

return True

else:

flag\_black\_attack = False

for x in range(10):

for y in range(10):

if self.deck\_model[x][y] == -1:

for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):

if 0 <= (x + i) <= 9 and 0 <= (y + j) <= 9:

if self.deck\_model[x + i][y + j] == 1 or self.deck\_model[x + i][y + j] == 2:

if 0 <= (x + i \* 2) <= 9 and 0 <= (y + j \* 2) <= 9:

if self.deck\_model[x + i + i][y + j + j] == 0:

flag\_black\_attack = True

coord\_checker.append([x, y, x + i + i, y + j + j])

elif self.deck\_model[x][y] == -2:

for d in range(4):

h = 0

if d == 0:

i = 1

j = 1

elif d == 1:

i = 1

j = -1

elif d == 2:

i = -1

j = 1

elif d == 3:

i = -1

j = -1

while 0 <= x + (i \* h) + i <= 9 and 0 <= y + (j \* h) + j <= 9:

if self.deck\_model[x + (i \* h)][y + (j \* h)] == 1 or \

self.deck\_model[x + (i \* h)][y + (j \* h)] == 2:

if self.deck\_model[x + (i \* h) + i][y + (j \* h) + j] == 0:

flag\_black\_attack = True

h += 1

while 0 <= x + (i \* h) <= 9 and 0 <= y + (j \* h) <= 9:

if self.deck\_model[x + (i \* h)][y + (j \* h)] == 0:

coord\_checker.append([x, y, x + (i \* h), y + (j \* h)])

else:

break

h += 1

else:

break

h += 1

if not flag\_black\_attack:

return True

else:

if coord\_checker\_user in coord\_checker:

return True

else:

return False

def multijump\_check(x2, y2):

if self.white\_move:

if self.deck\_model[x2][y2] == 1:

for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):

if 0 < (x2 + i) < 9 and 0 < (y2 + j) < 9:

if self.deck\_model[x2 + i][y2 + j] == -1 or self.deck\_model[x2 + i][y2 + j] == -2:

if 0 <= (x2 + i \* 2) <= 9 and 0 <= (y2 + j \* 2) <= 9:

if self.deck\_model[x2 + i + i][y2 + j + j] == 0:

return True

elif self.deck\_model[x2][y2] == 2:

for d in range(4):

if d == 0:

i = 1

j = 1

elif d == 1:

i = 1

j = -1

elif d == 2:

i = -1

j = 1

elif d == 3:

i = -1

j = -1

for h in range(1, 9):

if 0 < x2 + (i \* h) < 9 and 0 < y2 + (j \* h) < 9:

if self.deck\_model[x2 + (i \* h)][y2 + (j \* h)] == -1 or \

self.deck\_model[x2 + (i \* h)][y2 + (j \* h)] == -2:

if self.deck\_model[x2 + (i \* h) + i][y2 + (j \* h) + j] == 0:

return True

else:

break

else:

break

else:

if self.deck\_model[x2][y2] == -1:

for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):

if 0 < (x2 + i) < 9 and 0 < (y2 + j) < 9:

if self.deck\_model[x2 + i][y2 + j] == 1 or self.deck\_model[x2 + i][y2 + j] == 2:

if 0 <= (x2 + i \* 2) <= 9 and 0 <= (y2 + j \* 2) <= 9:

if self.deck\_model[x2 + i + i][y2 + j + j] == 0:

return True

elif self.deck\_model[x2][y2] == -2:

for d in range(4):

if d == 0:

i = 1

j = 1

elif d == 1:

i = 1

j = -1

elif d == 2:

i = -1

j = 1

elif d == 3:

i = -1

j = -1

for h in range(1, 9):

if 0 < x2 + (i \* h) < 9 and 0 < y2 + (j \* h) < 9:

if self.deck\_model[x2 + (i \* h)][y2 + (j \* h)] == 1 or \

self.deck\_model[x2 + (i \* h)][y2 + (j \* h)] == 2:

if self.deck\_model[x2 + (i \* h) + i][y2 + (j \* h) + j] == 0:

return True

else:

break

else:

break

return False

def move\_check(x1, y1, x2, y2):

self.move\_without\_attack = False

if self.deck\_model[x2][y2] != 0:

if self.white\_move:

messagebox.showwarning(title="Внимание", message="Такой ход невозможен")

return False

if self.deck\_model[x1][y1] == 2 or self.deck\_model[x1][y1] == -2:

if abs(x1 - x2) == abs(y1 - y2):

count\_checker\_queen = 0

coord\_checker = []

i = 0

j = 0

if x1 < x2:

i = 1

elif x2 < x1:

i = -1

if y1 < y2:

j = 1

elif y2 < y1:

j = -1

for h in range(1, abs(x2 - x1) + 1):

if self.deck\_model[x1 + (i \* h)][y1 + (j \* h)] == 1 or self.deck\_model[x1 + (i \* h)][y1 + (j \* h)] == 2:

if self.deck\_model[x1][y1] == 2:

if self.white\_move:

messagebox.showwarning(title="Внимание", message="Такой ход невозможен")

return False

else:

count\_checker\_queen += 1

coord\_checker = [x1 + (i \* h), y1 + (j \* h)]

if self.deck\_model[x1 + (i \* h)][y1 + (j \* h)] == -1 or self.deck\_model[x1 + (i \* h)][y1 + (j \* h)] == -2:

if self.deck\_model[x1][y1] == -2:

if self.white\_move:

messagebox.showwarning(title="Внимание", message="Такой ход невозможен")

return False

else:

count\_checker\_queen += 1

coord\_checker = [x1 + (i \* h), y1 + (j \* h)]

if count\_checker\_queen == 1:

self.deck\_model[coord\_checker[0]][coord\_checker[1]] = 0

self.deck\_model[x2][y2] = self.deck\_model[x1][y1]

self.deck\_model[x1][y1] = 0

return True

elif count\_checker\_queen > 1:

if self.white\_move:

messagebox.showwarning(title="Внимание", message="Такой ход невозможен")

return False

else:

if abs(x2 - x1) == 1 and abs(y2 - y1) == 1:

if self.black\_move:

if x2 < x1 and self.deck\_model[x1][y1] == -1:

return False

elif self.white\_move:

if x1 < x2 and self.deck\_model[x1][y1] == 1:

messagebox.showwarning(title="Внимание", message="Такой ход невозможен")

return False

if attack\_check(self.x1, self.y1, self.x2, self.y2):

self.deck\_model[x2][y2] = self.deck\_model[x1][y1]

self.deck\_model[x1][y1] = 0

self.move\_without\_attack = True

return True

elif abs(x2 - x1) == 2 and abs(y2 - y1) == 2:

if self.deck\_model[(x1 + x2) // 2][(y1 + y2) // 2] == 0 or self.deck\_model[(x1 + x2) // 2][(y1 + y2) // 2] == self.deck\_model[x1][y1]:

if self.white\_move:

messagebox.showwarning(title="Внимание", message="Такой ход невозможен")

return False

self.deck\_model[x2][y2] = self.deck\_model[x1][y1]

self.deck\_model[x1][y1] = 0

self.deck\_model[(x1 + x2) // 2][(y1 + y2) // 2] = 0

return True

if self.white\_move:

messagebox.showwarning(title="Внимание", message="Такой ход невозможен")

return False

def click\_event\_captured(event):

if self.white\_move:

if 0 < event.x < 500 and 0 < event.y < 500:

self.y2 = event.x // 50

self.x2 = event.y // 50

click2()

if self.black\_move:

directions = [[1, -1], [-1, 1], [-1, -1], [1, 1]]

direction = random.choice(directions)

self.x2 = direction[0]

self.y2 = direction[1]

click2()

def click2():

if self.first\_move:

if self.x1 == self.x2 and self.y1 == self.y2:

self.main.bind("<Button-1>", click\_event)

self.drawing\_move()

else:

if move\_check(self.x1, self.y1, self.x2, self.y2):

queen\_check(self.x2, self.y2)

self.drawing\_move()

if not self.move\_without\_attack:

if multijump\_check(self.x2, self.y2):

self.check\_end()

self.first\_move = False

self.x1, self.y1 = self.x2, self.y2

self.main.bind("<Button-1>", click\_event\_captured)

else:

self.white\_move, self.black\_move = not self.white\_move, not self.black\_move

self.check\_end()

self.user\_interact()

else:

self.white\_move, self.black\_move = not self.white\_move, not self.black\_move

self.check\_end()

self.user\_interact()

elif not self.first\_move:

if move\_check(self.x1, self.y1, self.x2, self.y2) and not self.move\_without\_attack:

queen\_check(self.x2, self.y2)

self.drawing\_move()

if multijump\_check(self.x2, self.y2):

self.check\_end()

self.x1, self.y1 = self.x2, self.y2

self.main.bind("<Button-1>", click\_event\_captured)

else:

self.drawing\_move()

self.first\_move = True

self.white\_move, self.black\_move = not self.white\_move, not self.black\_move

self.check\_end()

self.user\_interact()

def click\_event(event):

if self.white\_move:

if 0 < event.x < 500 and 0 < event.y < 500:

self.y1 = event.x // 50

self.x1 = event.y // 50

if self.deck\_model[self.x1][self.y1] == 1 or self.deck\_model[self.x1][self.y1] == 2:

self.deck.create\_image(self.y1 \* 50, self.x1 \* 50, anchor="nw", image=self.capture\_pic)

self.main.bind("<Button-1>", click\_event\_captured)

if self.black\_move :

coords = []

for x in range(10):

for y in range(10):

if self.deck\_model[x][y] == -1:

coords.append([x,y])

random\_checker = random.choice(coords)

self.x1 = random\_checker[0]

self.y1 = random\_checker[1]

if self.deck\_model[self.x1][self.y1] == -1 or self.deck\_model[self.x1][self.y1] == -2:

click\_event\_captured()

self.main.bind("<Button-1>", click\_event)

def check\_end(self):

flag\_white\_move = False

flag\_black\_move = False

for x in range(10):

for y in range(10):

if self.deck\_model[x][y] == 1 or self.deck\_model[x][y] == 2:

for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):

if 0 <= (x + i) <= 9 and 0 <= (y + j) <= 9:

if self.deck\_model[x + i][y + j] == -1 or self.deck\_model[x + i][y + j] == -2:

if 0 <= (x + i \* 2) <= 9 and 0 < (y + j \* 2) <= 9:

if self.deck\_model[x + i + i][y + j + j] == 0:

flag\_white\_move = True

break

elif self.deck\_model[x + i][y + j] == 0:

if self.deck\_model[x][y] == 1:

if x > x + i:

flag\_white\_move = True

break

else:

flag\_white\_move = True

break

elif self.deck\_model[x][y] == -1 or self.deck\_model[x][y] == -2:

for i, j in (1, -1), (-1, 1), (-1, -1), (1, 1):

if 0 <= (x + i) <= 9 and 0 <= (y + j) <= 9:

if self.deck\_model[x + i][y + j] == 1 or self.deck\_model[x + i][y + j] == 2:

if 0 <= (x + i \* 2) <= 9 and 0 <= (y + j \* 2) <= 9:

if self.deck\_model[x + i + i][y + j + j] == 0:

flag\_black\_move = True

break

elif self.deck\_model[x + i][y + j] == 0:

if self.deck\_model[x][y] == -1:

if x < x + i:

flag\_black\_move = True

break

else:

flag\_black\_move = True

break

if not flag\_white\_move:

self.end\_game("Черных")

elif not flag\_black\_move:

self.end\_game("Белых")

elif not flag\_white\_move and not flag\_black\_move:

self.end\_game("Ничья")

# Метод для вывода результатов окончания игры

def end\_game(self, winner):

def next\_game():

self.white\_move = True

self.black\_move = False

self.first\_move = True

self.deck\_model = [[0, -1, 0, -1, 0, -1, 0, -1, 0, -1],

[-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],

[0, -1, 0, -1, 0, -1, 0, -1, 0, -1],

[-1, 0, -1, 0, -1, 0, -1, 0, -1, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

[0, 1, 0, 1, 0, 1, 0, 1, 0, 1],

[1, 0, 1, 0, 1, 0, 1, 0, 1, 0],

[0, 1, 0, 1, 0, 1, 0, 1, 0, 1],

[1, 0, 1, 0, 1, 0, 1, 0, 1, 0]]

self.drawing\_move()

winner\_window = Tk()

winner\_window.geometry("150x150+150+150")

winner\_window.protocol('WM\_DELETE\_WINDOW', lambda: dismiss(winner\_window))

winner\_window.grab\_set()

if winner == "Ничья":

Label(winner\_window, text="Ничья").pack()

else:

Label(winner\_window, text=f"Победа {winner}").pack()

ttk.Button(winner\_window, text="Заново", command=lambda: (next\_game(), winner\_window.grab\_release(), winner\_window.destroy())).pack()

root = Tk()

root.title("Авторизация")

root.geometry("720x360-100-100")

icon = PhotoImage(file="checkers\_icon\_app.png")

root.iconphoto(True, icon)

root.resizable(False, False)

Checkers(root)

root.mainloop()