**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**на курсовую работу**

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

**Тема «Компьютерная логическая игра «Двухходовые шашки - Поддавки»»**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2401-49 ТЗ-3

Листов 22

**Руководитель разработки:**

доцент каф. ИВК, к.т.н., доцент

Шишкин Вадим Викторинович

« » 2023 г.

**Исполнитель:**

студент гр. ИСТбд-21

*Ефременко Станислав Алексеевич*

« » 202 г.

**2023**

**Содержание**

[**Аннотация** 3](#_Toc155678045)

[**ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 4](#_Toc155678046)

[**Пояснительная записка** 8](#_Toc155678047)

[**Руководство программиста** 15](#_Toc155678048)

[**Текст программы** 23](#_Toc155678049)

# **Аннотация**

Данный документ представляет собой пояснительную записку на курсовую работу на тему «Компьютерная логическая игра «Двухходовые шашки - Поддавки»». Документ содержит следующие разделы: техническое задание, пояснительная записка и руководство программиста, код программы; в нем излагается постановка задачи и описание реализуемой программы, ее назначение. Документ может быть использован в качестве инструкции для применения рассматриваемого программного средств

**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 «Двухходовые шашки - Поддавки»»**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2401-49 ТЗ-3

Листов 4

**Исполнитель:**

студент гр. ИСТбд-21

*Ефременко Станислав Алексеевич*

« » 202 г.

**2023**

**Введение**

Компьютерная логическая игра «Двухходовые шашки – Поддавки».

Эта игра ведется по правилам Русских шашек, за исключением одного важного добавления. Ход игрока состоит из двух полуходов. При этом, если в результате выполненного первого полухода появляется возможность рубки, то выполнять ее обязательно. Если рубка возможна перед первым полуходом, то он также обязателен. Выполнение двух полуходов обязательно. Если игрок не может выполнить два полухода, то он считается победителем.

Шашечная доска - 64-клеточная (8×8) доска, располагается между партнерами таким образом, чтобы слева от играющего находилось тёмное угловое поле. В начальной позиции у каждой стороны по 12 шашек, которые занимают первые три ряда с каждой стороны. Игра ведется по тёмным полям. Шашки делятся на простые и дамки. В начальном положении все шашки простые. При достижении последнего (восьмого от себя) горизонтального ряда простая шашка превращается в дамку.

**Правила игры**

• Простая шашка ходит по диагонали вперёд на одну клетку.

• Дамка ходит по диагонали на любое свободное поле как вперёд, так и назад.

• Взятие обязательно. Побитые шашки и дамки снимаются только после завершения хода.

• Простая шашка, находящаяся рядом с шашкой соперника, за которой имеется свободное поле, переносится через эту шашку на это свободное поле. Если есть возможность продолжить взятие других шашек соперника, то это взятие продолжается, пока бьющая шашка не достигнет положения, из которого бой невозможен. Взятие простой шашкой производится как вперёд, так и назад.

• Дамка бьёт по диагонали, как вперёд, так и назад, и становится на любое свободное поле после побитой шашки. Аналогично, дамка может бить несколько фигур соперника и должна бить до тех пор, пока это возможно.

• При бое через дамочное поле простая шашка превращается в дамку и продолжает бой по правилам дамки.

• При взятии применяется правило турецкого удара — за один ход шашку противника можно побить только один раз. То есть, если при бое нескольких шашек противника шашка или дамка повторно выходит на уже побитую шашку, то ход останавливается.

• При нескольких вариантах взятия, например, одну шашку или две, игрок выбирает вариант взятия по своему усмотрению.

• Выигрывает тот, у кого не осталось ни одной шашки, или оставшиеся шашки не могут сделать ни одного хода.

**Партия считается закончившейся вничью в следующих случаях:**

• Если один из участников предлагает ничью, а другой её принимает;

• При невозможности выигрыша ни одного из соперников;

• Если три раза повторяется одна и та же позиция;

• Если участник, имеющий три дамки (и более) против одной дамки противника, за 15 ходов не возьмёт дамку противника;

• Если в позиции, в которой оба соперника имеют дамки, не изменилось соотношение сил (то есть не было взятия, и ни одна простая шашка не стала дамкой) на протяжении:

• в 2-х и 3-х фигурных окончаниях — 5 ходов,

• в 4-х и 5-и фигурных окончаниях — 30 ходов,

• в 6-и и 7-и фигурных окончаниях — 60 ходов;

• Если участник, имея в окончании партии три дамки, две дамки и простую, дамку и две простые, три простые против одинокой дамки, находящейся на большой дороге, своим 5-м ходом не сможет добиться выигранной позиции;

• Если в течение 15 ходов игроки делали ходы только дамками, не передвигая простых шашек и не производя взятия.

**функциональные возможности**

Должно отображение поле игры. Шашки должны передвигаться по правилам игры. Должно быть окно авторизации/регистрации с возможностью сохранения пользователей.

**1. Основания для разработки**

Разработка основана на учебном плане направления 09.03.02 «Информационные системы и технологии» и распоряжения по факультету.

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

**2.1. Функциональное назначение**

Функциональное назначение игры «канадские шашки» - предоставление приложения для игры в канадские шашки. Основные автоматизируемые процессы включают в себя отображение игрового поля, размещение и перемещение шашек, проверку правильности ходов, определение победы, поражения. Группа пользователей: игроки, которые могут управлять шашками на игровом поле.

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

Приложение должно быть разработано в виде одного модуля с дополнительными информационными файлами при необходимости.

2.2.2 Требования к составу функций приложения

Наиболее важными функциями в приложении являются, функции авторизации/регистрации, функция отрисовки игрового поля и отрисовки шашек при ходе, функции отвечающие за проверку правильности хода и воспроизведения заданного хода, а также функция случайного хода компьютера.

2.2.3 Требования к организации информационного обеспечения, входных и выходных данных

В приложении должен быть реализован графический интерфейс взаимодействия с пользователем. Изображения шашек могут храниться в отдельных графических файлах. Логин и пароль пользователя должны вводиться с клавиатуры. Логины и пароли зарегистрированных пользователей должны храниться в отдельном файле или базе данных в зашифрованном виде. Пояснительные информационные сообщения для пользователя должны выводиться внизу игрового поля по ходу игры.

**2.3 Требования к надёжности**

Приложение должно быть стабильным и работоспособным, не вызывать сбоев или ошибок. В случае сбоя или ошибки, приложение должно быть способно восстановиться без потери данных.

**2.4 Требования к информационной и программной совместимости**

Операционная система: Windows 11 Домашняя.

Версия языка программирования: Python 3.11.3.

Среда разработки: PyCharm – 2022.2.2.

При создании программы используются встроенные библиотеки “os”, “json”, “copy”, “random” и всторонняя библиотека “tkinter 8.6.”.

**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

Проект будет храниться в репозитории на сайте github.com по ссылке https://github.com/noxe19/coursework.

2.6 3 Сроки хранения

Срок хранения – до июля 2023 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 «Двухходовые шашки - Поддавки»»**

# **Пояснительная записка**

Р.02069337.22/2401-49 ТЗ-3

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Листов 7

**Исполнитель**:

студент гр. ИСТбд-21

*Ефременко Станислав Алексеевич*

« » 2023 г.

**2023**

**Введение**

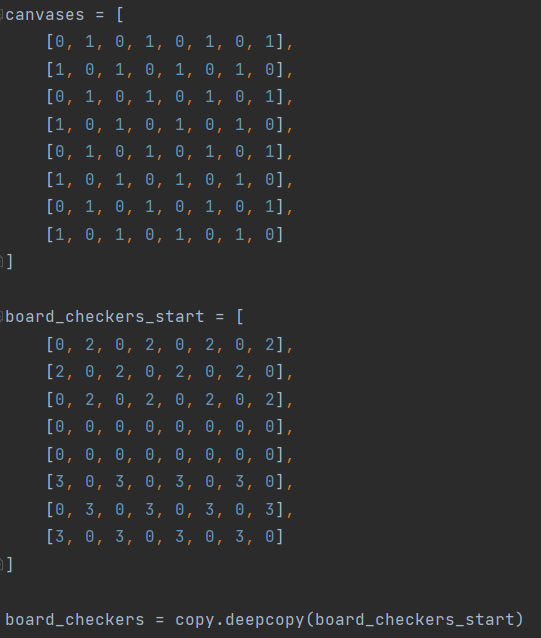
Компьютерная логическая игра «Двухходовые шашки – Поддавки». Приложение было разработано на языке Python 3.11.3 используя функциональный подход, этот подход был выбран на основании полученных знаний за период обучения. Были реализованы алгоритмы в виде отдельных функций, которые работают с данными и взаимодействуют между собой.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу.

**1.2 Математические методы**



Матрица canvases описывает расположение черных и белых клеток (1 – черные, 0 - белые)

Матрица board\_checkers описывает расположение шашек на доске во время игры

Матрица board\_checkers\_start описывает начальное расположение шашек на доске

* 0 – пустая клетка
* 2 – красная шашка
* 3 – белая шашка
* 12 – красная дамка
* 13 – белая дамка

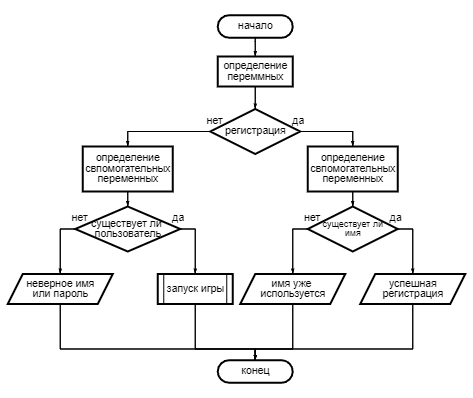
Выбор такой модели обусловлен ее эффективностью и простотой в использовании.

**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура

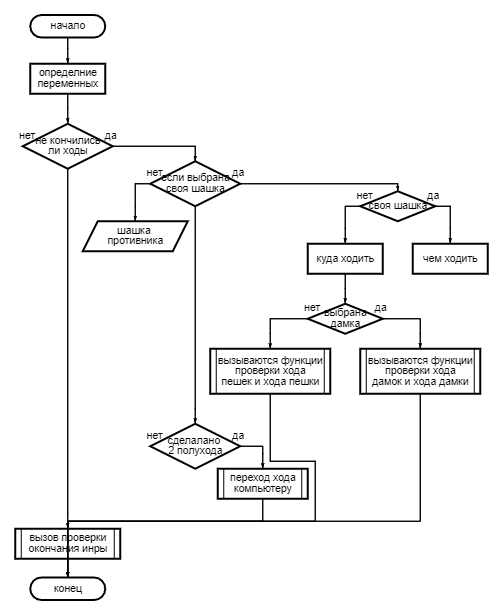
В приложении были использованы такие структуры данных как: массивы и словари. Функции взаимодействуют между собой.

1.3.2. Алгоритм авторизации/регистрации



Данный алгоритм производит регистрацию и авторизацию пользователя, при корректном вводе данных.

1.3.2. Алгоритм хода игрока



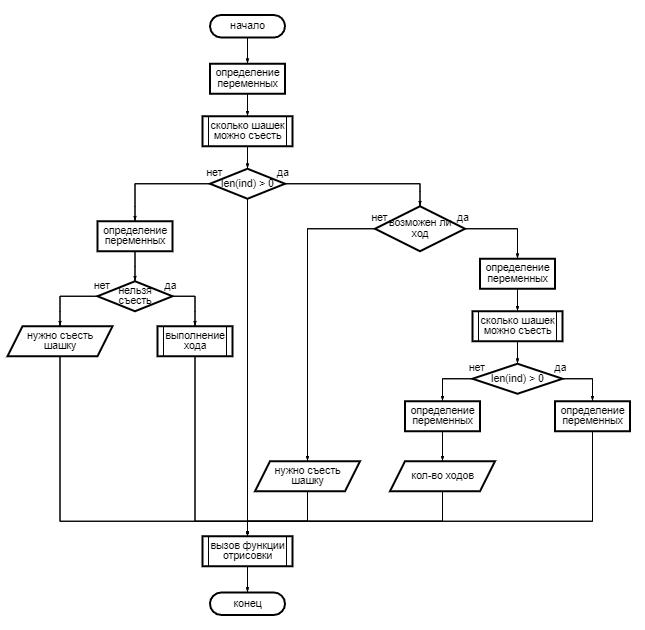
Данный алгоритм производит ход игрока на основе данных полученных после проверки возможности хода, в приоритете съедание шашки противника.

1.3.2. Алгоритм проверки ходов дамок



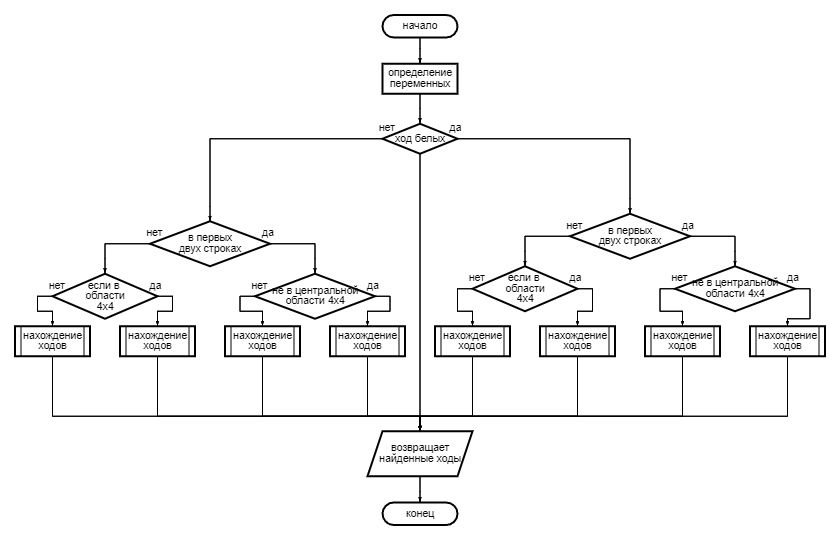
Данный алгоритм просчитывает все возможные ходы дамок.

1.3.2. Алгоритм хода дамки



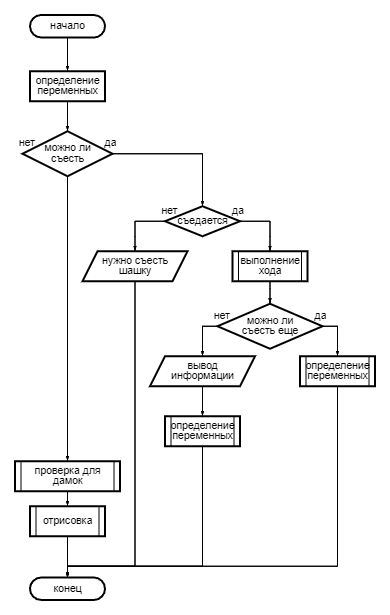
Данный алгоритм производит ход дамки, в приоритете съедание шашки противника.

1.3.2. Алгоритм проверки хода пешки



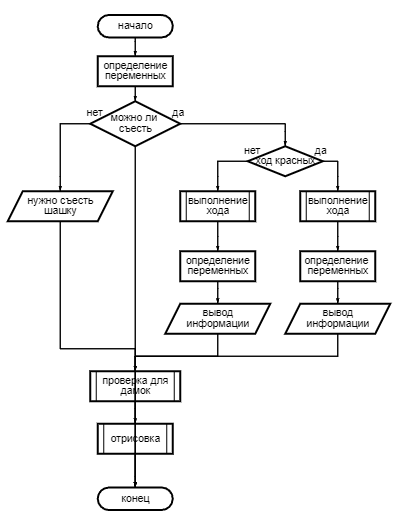
Данный алгоритм проверяет ход для пешки, вычисляет возможно ли съесть шашку противника.

1.3.2. Алгоритм хода пешки при съедании вражеской



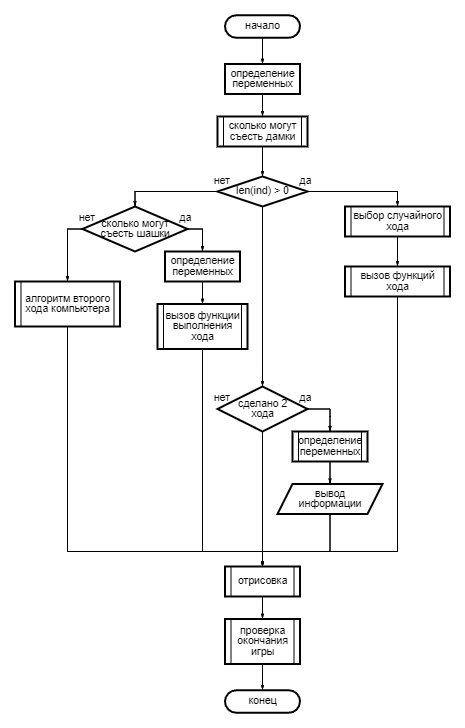
Данный алгоритм воспроизводит пешки при съедании вражеской пешки.

1.3.2. Алгоритм хода пешки на одну клетку



Данный алгоритм воспроизводит ход пешки на одну клетку.

1.3.2. Алгоритм хода противника



Данный алгоритм воспроизводит случайный ход компьютера, на основе данных полученных функциями проверки ходов пешек и дамок.

**1.4 Тестирование**

Весь процесс тестирования проходил вручную, без привлечения специального ПО. На протяжении всего хода разработки, использовался метод белого ящика, так как в любом время имелся доступ ко всем компонентам программы. Всё тестирование выполнялось интуитивным методом, без подготовки специальных тестов.

На протяжении всего хода разработки, по мере добавления новых функций программы, использовалось системное тестирование новых функций, для устранения возникших в ходе написания ошибок. После положительных результатов тестирования функция считалась внедренной.

**2. Источники, использованные при разработке**

1. Правила игры [Электронный ресурс]: Русские шашки – URL: https://lotos-khv.ru/game/games/russian\_checkers.pdf (дата обращения: 16.12.2023)
2. Правила игры [Электронный ресурс]: Двухходовые шашки – URL:

https://lotos-khv.ru/game/games/dvuxxod.pdf (дата обращения: 16.12.2023)

1. Шишкин, В.В. Разработка логических компьютерных игр с графическим интерфейсом в среде Python [Электронный ресурс] / В.В. Шишкин, Д.С. Афонин. – Ульяновск: УлГТУ, 2023. – 89 с. – Режим доступа: для всех пользователей. – URL: <http://lib.ulstu.ru/venec/disk/2023/112.pdf> (дата обращения: 16.11.23).

**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 «Двухходовые шашки - Поддавки»»

# **Руководство программиста**

Р.02069337.22/2401-49 ТЗ-3

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Листов 8

**Исполнитель:**

студент гр. ИСТбд-21

*Ефременко Станислав Алексеевич*

« » 2023 г.

**2023**

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложения предназначено для игры в компьютерную логическую игру «Двухходовые шашки – Поддавки».

Правила игры

• Простая шашка ходит по диагонали вперёд на одну клетку.

• Дамка ходит по диагонали на любое свободное поле как вперёд, так и назад.

• Взятие обязательно. Побитые шашки и дамки снимаются только после завершения хода.

• Простая шашка, находящаяся рядом с шашкой соперника, за которой имеется свободное поле, переносится через эту шашку на это свободное поле. Если есть возможность продолжить взятие других шашек соперника, то это взятие продолжается, пока бьющая шашка не достигнет положения, из которого бой невозможен. Взятие простой шашкой производится как вперёд, так и назад.

• Дамка бьёт по диагонали, как вперёд, так и назад, и становится на любое свободное поле после побитой шашки. Аналогично, дамка может бить несколько фигур соперника и должна бить до тех пор, пока это возможно.

• При бое через дамочное поле простая шашка превращается в дамку и продолжает бой по правилам дамки.

• При взятии применяется правило турецкого удара — за один ход шашку противника можно побить только один раз. То есть, если при бое нескольких шашек противника шашка или дамка повторно выходит на уже побитую шашку, то ход останавливается.

• При нескольких вариантах взятия, например, одну шашку или две, игрок выбирает вариант взятия по своему усмотрению.

• Выигрывает тот, у кого не осталось ни одной шашки, или оставшиеся шашки не могут сделать ни одного хода.

Функциональные возможности, которые предоставляет приложение:

* Авторизация/регистрация
* Выполнять ход пешкой/дамкой
* Съедать вражеские шашки
* Проиграть или выйграть

**1.2 Условия, необходимые для использования приложения**

Операционная система: Windows 11 Домашняя.

Версия языка программирования: Python 3.11.3.

Среда разработки: PyCharm – 2022.2.2.

При создании программы используются встроенные библиотеки “os”, “json”, “copy”, “random” и “tkinter 8.6.”.

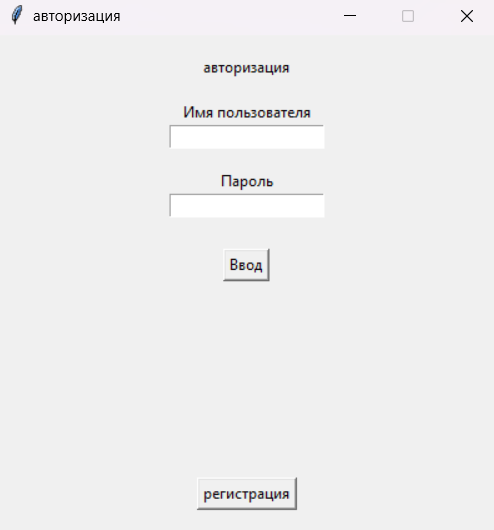
**2. Характеристики программы**

**2.1 Характеристики приложения**

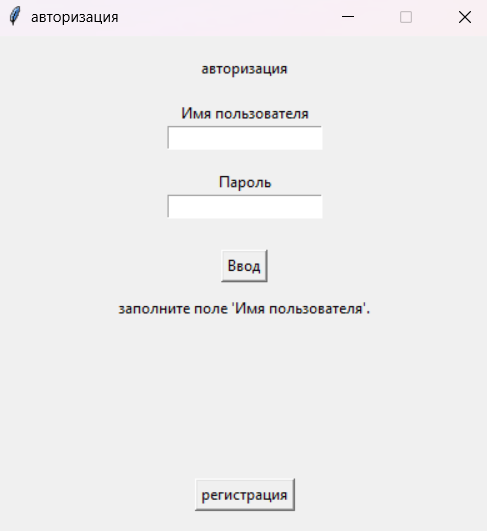
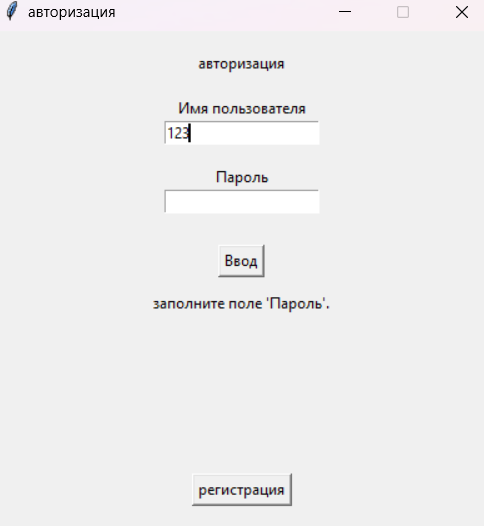
Значимых строк программного кода - 802. Количество структур - 54, Количество алгоритмов - 24.

“os” – используется для работы с файлами, “json” – используется для работы с json фалами, “copy” – используется для глубоко копирования (значения, а не ссылки на значение), “random” – используется для получения случайныз чисел, чтобы реализовать случайные ходы компьютера и “tkinter 8.6.” – используется для создания интерфейса.

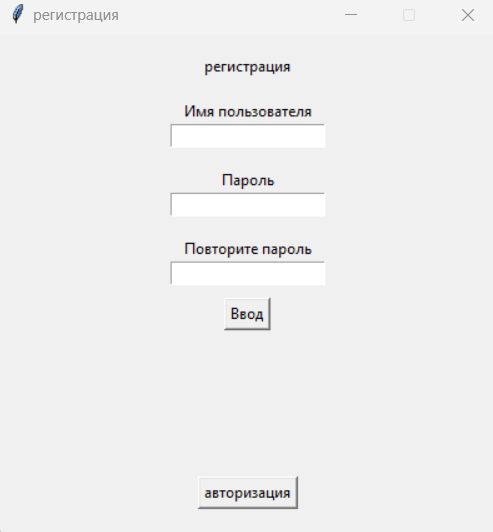
При запуске открывается окно регистрации, где нужно заполнить поля: «Имя пользователя», «Пароль».



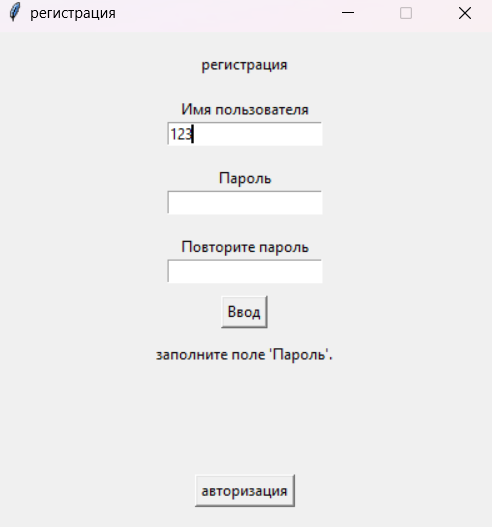
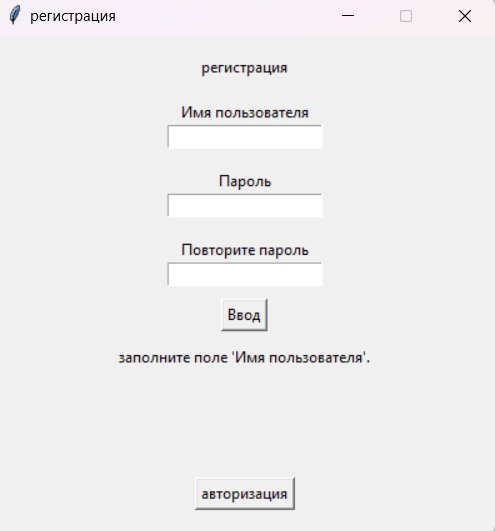
При нажатии на кнопку ввод появляется уведомление о текущем состоянии.

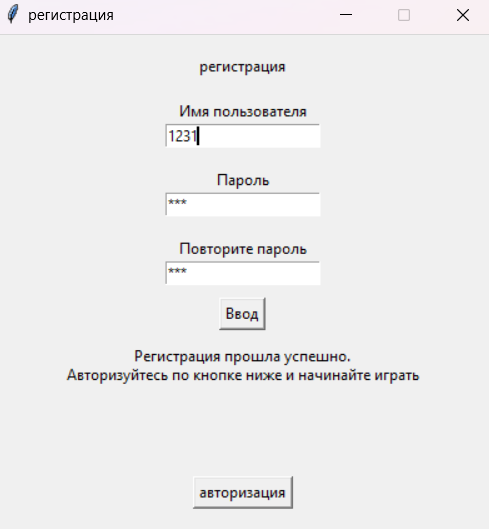
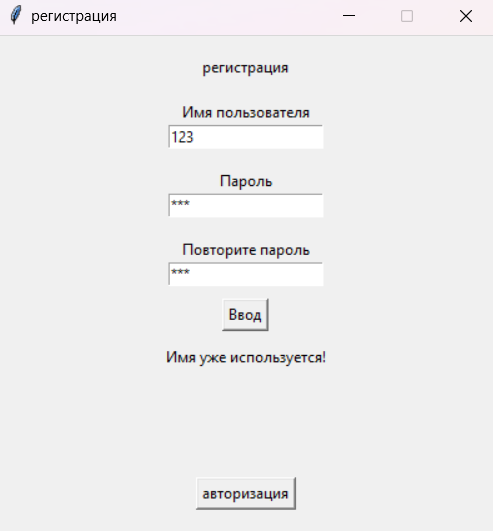
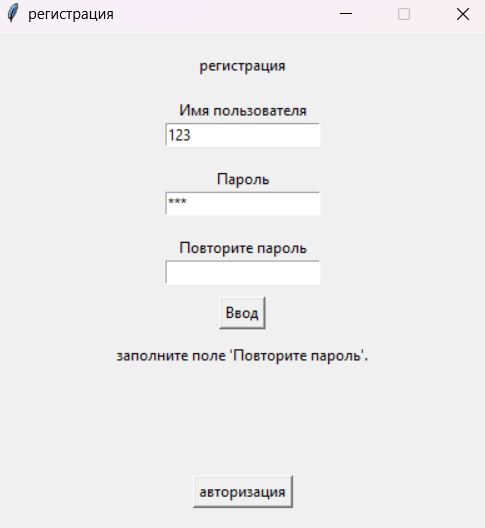
 

При нажатии на кнопку «регистрация», закрывается окно регистрации и открывается окно регистрации, где нужно заполнить поля: «Имя пользователя», «Пароль», «Повторите пароль»

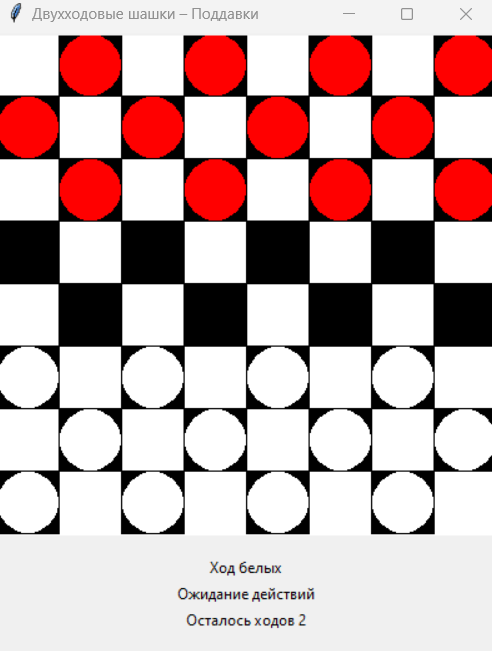


При нажатии на кнопку ввод появляется уведомление о текущем состоянии

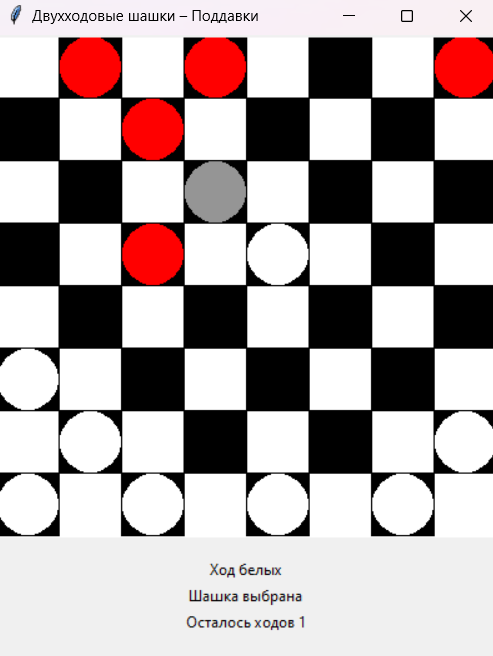




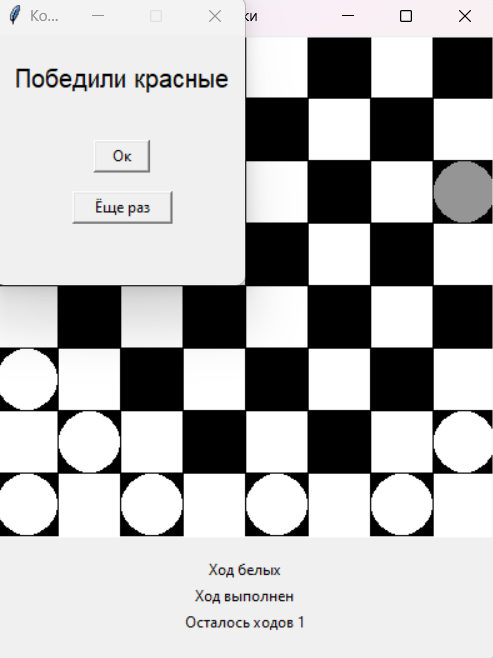
После успешной авторизации открывается окно с игрой. Состоит из основного (игрового) поля и поля для вывода информации (снизу). В поле информации выводится текущее состояние игры



На поле есть шашки и дамки, которыми управляет игрок



В конце игры появляется окно информирующее о победителе.



Кнопка «ОК» - при нажатии закрывает приложение, кнопка «Еще раз» - начинает игру сначала.

Производится проверка, на наличие пустой строки в полях ввода.

**2.2 Особенности реализации приложения**

В приложении были использованы такие структуры данных как: массивы и словари. Вместо этих структур можно было использовать списки, но были выбраны массивы и словари из-за их гибкости и относительной простоты при использовании.

**3. Обращение к программе**

Использованные методы и алгоритмы:

1. user\_file – создание или открытие файла, чтобы занести в него нового пользователя или запустить игру при авторизации польхователя
2. read\_user\_file – построчно считывает данные о пользователях из файла, для дальнейшей работы с ними, и заносит их в массив
3. encrypt\_caesar – шифрование логина и пароля пользователя
4. board – создание окна игры
5. click – обработчик кликов для игрового поля
6. register – создание окна регистрации
7. input\_click\_register – обработка введенных данных на корректность и передача их в функции работы с файлом
8. register\_click\_register – закрытие окна регистрации и создание окна авторизации
9. login\_fun – создание окна авторизации
10. input\_click\_login – обработка введенных данных на корректность и передача их в функции работы с файлом
11. register\_click\_login – закрытие окна авторизации и создание окна регистрации
12. board\_rendering – отрисовка игрового поля и шашек
13. turn\_player\_click – функция хода игрока
14. killed\_checker – промежуточная функция для определения нужной функции хода
15. queen\_check – проверяет возможные ходы дамок
16. queen\_move – ход дамки
17. moved – съедание пешки противника
18. one\_move – функция одного хода для пешки
19. become\_queen – делает из пешек дамки, если они попали в дамочную область
20. check\_kill – проверяет, может ли пешка съесть вражескую
21. minimax\_one\_move – функция для хода компьютера на одну клетку
22. minimax – функция для хода компьютера
23. check\_win – выявление победителя
24. close\_game – полностью закрывает приложение
25. game\_restart – пересоздает поле и шашки на доске

Использованные библиотеки:

1. Tkinter — это кроссплатформенный графический интерфейс Python, позволяющий работать с библиотекой Tk. Он содержит элементы графического интерфейса пользователя, с помощью которых можно создавать различные приложения.
2. Json - предоставляет методы для кодирования и декодирования данных в формате JSON.
3. Copy - предоставляет функции для копирования объектов.
4. Os - предоставляет функции для взаимодействия с операционной системой. Эта библиотека позволяет выполнять различные операции, такие как работа с файлами и каталогами, управление переменными окружения, выполнение команд в командной строке и многое другое.
5. Random - предоставляет функции для генерации случайных чисел и выполнения различных операций, связанных с случайностью.

**4. Сообщения**

При корректном вводе в интерфейсе выводятся сообщения:  
Регистрация прошла успешно. Авторизуйтесь по кнопке ниже и начинайте играть.

* Имя уже используется!
* Неверное имя пользователя или пароль, повторите попытку или зарегистрируйтесь, нажав на кнопку ниже.
* Пароли не совпадают!

При не корректном вводе в интерфейсе выводятся сообщения:

* заполните поле 'Имя пользователя'.
* заполните поле 'Пароль'.
* заполните поле 'Повторите пароль'.

При корректном выводе в всплывающем окне выводятся сообщения:

* Победили красные
* Победили белые

# **Текст программы**

from tkinter import \*  
import json  
import copy  
import os  
from random import randint  
  
  
click1 = [0, 0]  
click2 = [0, 0]  
click\_buf = 0  
player\_move = 3  
move\_count = 0  
click\_stop = True  
shift\_amount = 3  
  
canvases = [  
 [0, 1, 0, 1, 0, 1, 0, 1],  
 [1, 0, 1, 0, 1, 0, 1, 0],  
 [0, 1, 0, 1, 0, 1, 0, 1],  
 [1, 0, 1, 0, 1, 0, 1, 0],  
 [0, 1, 0, 1, 0, 1, 0, 1],  
 [1, 0, 1, 0, 1, 0, 1, 0],  
 [0, 1, 0, 1, 0, 1, 0, 1],  
 [1, 0, 1, 0, 1, 0, 1, 0]  
]  
  
board\_checkers\_start = [  
 [0, 2, 0, 2, 0, 2, 0, 2],  
 [2, 0, 2, 0, 2, 0, 2, 0],  
 [0, 2, 0, 2, 0, 2, 0, 2],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [0, 0, 0, 0, 0, 0, 0, 0],  
 [3, 0, 3, 0, 3, 0, 3, 0],  
 [0, 3, 0, 3, 0, 3, 0, 3],  
 [3, 0, 3, 0, 3, 0, 3, 0]  
]  
  
board\_checkers = copy.deepcopy(board\_checkers\_start)  
  
  
def user\_file(user\_name, password, shift\_amount, mode, lab\_erorr, main):  
 file\_path = 'user.json'  
 data = {  
 "user\_name": encrypt\_caesar(user\_name, shift\_amount),  
 "password": encrypt\_caesar(password, shift\_amount),  
 }  
 if mode:  
 buf = True  
 user\_data = read\_user\_file()  
 for user in user\_data:  
 if user["user\_name"] == data["user\_name"]:  
 buf = False  
 break  
 if buf:  
 with open(file\_path, 'a') as file:  
 json.dump(data, file)  
 file.write('\n')  
 lab\_erorr.configure(text="Регистрация прошла успешно.\nАвторизуйтесь по кнопке ниже и начинайте играть")  
 else:  
 lab\_erorr.configure(text="Имя уже используется!")  
 else:  
 buf = True  
 user\_data = read\_user\_file()  
 for user in user\_data:  
 if user["user\_name"] == data["user\_name"] and user["password"] == data["password"]:  
 buf = False  
 break  
 if buf:  
 lab\_erorr.configure(  
 text="Неверное имя пользователя или пароль, "  
 "\nповторите попытку или зарегистрируйтесь, \nнажав на кнопку ниже.")  
 else:  
 main.destroy()  
 board()  
  
  
def read\_user\_file():  
 file\_path = 'user.json'  
 users = []  
 if os.path.exists(file\_path):  
 with open(file\_path, 'r') as json\_file:  
 for line in json\_file:  
 data = json.loads(line.strip())  
 users.append(data)  
 return users  
  
  
def encrypt\_caesar(plaintext, shift):  
 ciphertext = ""  
 for char in plaintext:  
 shifted\_char = chr(ord(char) + shift)  
 ciphertext += shifted\_char  
 return ciphertext  
  
  
def board():  
 global game\_window  
 game\_window = Tk()  
 game\_window.title("Двухходовые шашки – Поддавки")  
 game\_window.geometry('400x500+100+100')  
 game\_window.resizable(width=True, height=False)  
 game\_window.minsize(400, 500)  
 global board\_canvas  
 board\_canvas = Canvas(game\_window, bg="white", width=400, height=400)  
 board\_canvas.bind("<Button-1>", click)  
 board\_canvas.pack(side=TOP)  
 board\_rendering()  
 global lbl\_count  
 lbl\_count = Label(game\_window, text=f'Осталось ходов {2 - move\_count}')  
 lbl\_count.pack(side=BOTTOM, pady=(0, 20))  
 global lbl\_info  
 lbl\_info = Label(game\_window, text='Ожидание действий')  
 lbl\_info.pack(side=BOTTOM)  
 global lbl\_player  
 lbl\_player = Label(game\_window, text='Ход белых')  
 lbl\_player.pack(side=BOTTOM)  
 mainloop()  
  
  
def click(event):  
 x = event.x // 50  
 y = event.y // 50  
 board\_rendering()  
 if player\_move == 3:  
 turn\_player\_click(x, y)  
 else:  
 minimax()  
  
  
def register():  
 main = Tk()  
 main.geometry('400x400+100+100')  
 main.resizable(width=False, height=False)  
 main.title('регистрация')  
 lab = Label(text='регистрация')  
 lab\_name = Label(text='Имя пользователя')  
 ent\_login = Entry()  
 lab\_password = Label(text='Пароль')  
 ent\_password = Entry(show='\*')  
 lab\_repeat\_password = Label(text='Повторите пароль')  
 ent\_repeat\_password = Entry(show='\*')  
 lab\_erorr = Label(text="")  
 btn\_input = Button(text='Ввод', command=lambda window=main,login=ent\_login, password=ent\_password, repeat\_password=ent\_repeat\_password, error=lab\_erorr: input\_click\_register(main, login, password, repeat\_password, error))  
 btn\_register = Button(text='авторизация', command=lambda window1=main: register\_click\_register(window1))  
 lab.pack(pady=15)  
 lab\_name.pack()  
 ent\_login.pack(pady=(0, 15))  
 lab\_password.pack()  
 ent\_password.pack(pady=(0, 15))  
 lab\_repeat\_password.pack()  
 ent\_repeat\_password.pack()  
 btn\_input.pack(pady=10)  
 lab\_erorr.pack()  
 btn\_register.pack(side=BOTTOM, pady=20)  
 main.mainloop()  
  
  
def input\_click\_register(main, ent\_login, ent\_password, ent\_repeat\_password, lab\_erorr):  
 name = ent\_login.get()  
 password = ent\_password.get()  
 repeat\_password = ent\_repeat\_password.get()  
 if name == '':  
 lab\_erorr.configure(text="заполните поле 'Имя пользователя'.")  
 elif password == '':  
 lab\_erorr.configure(text="заполните поле 'Пароль'.")  
 elif repeat\_password == '':  
 lab\_erorr.configure(text="заполните поле 'Повторите пароль'.")  
 else:  
 if password != repeat\_password:  
 lab\_erorr.configure(text="Пароли не совпадают!")  
 else:  
 user\_file(name, password, shift\_amount, True, lab\_erorr, main)  
  
  
def register\_click\_register(main):  
 main.destroy()  
 login\_fun()  
  
  
def login\_fun():  
 main = Tk()  
 main.geometry('400x400+100+100')  
 main.resizable(width=False, height=False)  
 main.title('авторизация')  
 lab = Label(text='авторизация')  
 lab\_name = Label(text='Имя пользователя')  
 ent\_login = Entry()  
 lab\_password = Label(text='Пароль')  
 ent\_password = Entry(show='\*')  
 lab\_erorr = Label(text="")  
 btn\_input = Button(text='Ввод', command=lambda window1=main, login=ent\_login, password=ent\_password, error=lab\_erorr: input\_click\_login(window1, login, password, error))  
 btn\_register = Button(text='регистрация', command=lambda window1=main: register\_click\_login(window1))  
 lab.pack(pady=15)  
 lab\_name.pack()  
 ent\_login.pack(pady=(0, 15))  
 lab\_password.pack()  
 ent\_password.pack(pady=(0, 15))  
 btn\_input.pack(pady=10)  
 lab\_erorr.pack()  
 btn\_register.pack(side=BOTTOM, pady=20)  
 main.mainloop()  
  
  
def input\_click\_login(main, ent\_login, ent\_password, lab\_erorr):  
 name = ent\_login.get()  
 password = ent\_password.get()  
 if name == '':  
 lab\_erorr.configure(text="заполните поле 'Имя пользователя'.")  
 elif password == '':  
 lab\_erorr.configure(text="заполните поле 'Пароль'.")  
 else:  
 user\_file(name, password, shift\_amount, False, lab\_erorr, main)  
  
  
def register\_click\_login(main):  
 main.destroy()  
 register()  
  
  
def board\_rendering():  
 board\_canvas.delete("all")  
 for i in range(8):  
 for j in range(8):  
 if (i + j) % 2 == 1:  
 board\_canvas.create\_rectangle(j \* 50, i \* 50, (j + 1) \* 50, (i + 1) \* 50, fill="black", outline="black")  
 if board\_checkers[i][j] == 2:  
 board\_canvas.create\_oval(j \* 50, i \* 50, (j + 1) \* 50, (i + 1) \* 50, fill="red", outline="black")  
 if board\_checkers[i][j] == 3:  
 board\_canvas.create\_oval(j \* 50, i \* 50, (j + 1) \* 50, (i + 1) \* 50, fill="white", outline="black")  
 if board\_checkers[i][j] == 12:  
 board\_canvas.create\_oval(j \* 50, i \* 50, (j + 1) \* 50, (i + 1) \* 50, fill="#A20000", outline="black")  
 if board\_checkers[i][j] == 13:  
 board\_canvas.create\_oval(j \* 50, i \* 50, (j + 1) \* 50, (i + 1) \* 50, fill="#959595", outline="black")  
  
  
def turn\_player\_click(x, y):  
 global board\_checkers  
 global player\_move  
 global move\_count  
 global click\_buf  
 global click1  
 global click2  
 if move\_count < 2:  
 if board\_checkers[y][x] != 2:  
 if board\_checkers[y][x] == 3 or board\_checkers[y][x] == 13:  
 if click\_stop:  
 click1[0] = y  
 click1[1] = x  
 click\_buf = 1  
 lbl\_info.configure(text="Шашка выбрана")  
 elif board\_checkers[y][x] == 0:  
 if canvases[y][x] == 1:  
 click2[0] = y  
 click2[1] = x  
 click\_buf = 0  
 if board\_checkers[click1[0]][click1[1]] == 13:  
 moves, kills, move = queen\_check()  
 queen\_move(kills, moves, move)  
 else:  
 kill1, kill2, kill3 = check\_kill(3)  
 killed\_checker(kill1, kill2, kill3)  
 else:  
 lbl\_info.configure(text="Выберите другой ход")  
 else:  
 lbl\_info.configure(text="Это шашка соперника")  
 if move\_count == 2:  
 move\_count = 0  
 player\_move = 2  
 lbl\_player.configure(text="Ход красных")  
 lbl\_count.configure(text=f"Осталось ходов {2 - move\_count}")  
 check\_win()  
 minimax()  
 check\_win()  
  
  
def killed\_checker(kill1, kill2, kill3):  
 global move\_count  
 global board\_checkers  
 if len(kill1) > 0:  
 if abs(click2[0] - click1[0]) == 2 and abs(click2[1] - click1[1]) == 2:  
 moved(kill1, kill2, kill3)  
 else:  
 lbl\_info.configure(text="Нужно съесть вражескую шашку")  
 elif len(kill1) == 0:  
 if abs(click2[0] - click1[0]) == 1 and abs(click2[1] - click1[1]) == 1:  
 one\_move()  
 else:  
 print("сходите в другое место")  
  
  
def queen\_check():  
 kill = []  
 move = []  
 kills = []  
 moves = []  
 buf\_move = []  
 k = -1  
 p = player\_move + 10  
 if p == 12:  
 p\_queen1, p1, p\_queen2, p2 = 12, 2, 13, 3  
 else:  
 p\_queen1, p1, p\_queen2, p2 = 13, 3, 12, 2  
 for y in range(8):  
 for x in range(8):  
 buf\_x = x  
 buf\_y = y  
 if board\_checkers[buf\_y][buf\_x] == p\_queen1:  
 buf\_move.append([buf\_y, buf\_x])  
 kills.append([])  
 moves.append([])  
 k += 1  
 while buf\_y > 0 and buf\_x > 0:  
 if board\_checkers[buf\_y - 1][buf\_x - 1] == p1 or board\_checkers[buf\_y - 1][buf\_x - 1] == p\_queen1:  
 break  
 elif buf\_y > 1 and buf\_x > 1:  
 if board\_checkers[buf\_y - 1][buf\_x - 1] == p2 or board\_checkers[buf\_y - 1][buf\_x - 1] == p\_queen2:  
 if board\_checkers[buf\_y - 2][buf\_x - 2] == 0:  
 move = []  
 kill.append([buf\_y - 1, buf\_x - 1])  
 move.append([buf\_y - 2, buf\_x - 2])  
 buf\_y -= 1  
 buf\_x -= 1  
 while buf\_y > 0 and buf\_x > 0:  
 if board\_checkers[buf\_y - 1][buf\_x - 1] == 0:  
 move.append([buf\_y - 1, buf\_x - 1])  
 else:  
 break  
 buf\_y -= 1  
 buf\_x -= 1  
 if board\_checkers[buf\_y - 1][buf\_x - 1] == 0:  
 move.append([buf\_y - 1, buf\_x - 1])  
 else:  
 break  
 elif board\_checkers[buf\_y - 1][buf\_x - 1] == 0:  
 move.append([buf\_y - 1, buf\_x - 1])  
 else:  
 if board\_checkers[buf\_y - 1][buf\_x - 1] == 0:  
 move.append([buf\_y - 1, buf\_x - 1])  
 buf\_y -= 1  
 buf\_x -= 1  
 kills[k].append(kill)  
 moves[k].append(move)  
 kill = []  
 move = []  
 buf\_x = x  
 buf\_y = y  
 while buf\_y > 0 and buf\_x < 7:  
 if board\_checkers[buf\_y - 1][buf\_x + 1] == p1 or board\_checkers[buf\_y - 1][buf\_x + 1] == p\_queen1:  
 break  
 elif buf\_y > 1 and buf\_x < 6:  
 if board\_checkers[buf\_y - 1][buf\_x + 1] == p2 or board\_checkers[buf\_y - 1][buf\_x + 1] == p\_queen2:  
 if board\_checkers[buf\_y - 2][buf\_x + 2] == 0:  
 move = []  
 kill.append([buf\_y - 1, buf\_x + 1])  
 move.append([buf\_y - 2, buf\_x + 2])  
 buf\_y -= 1  
 buf\_x += 1  
 while buf\_y > 0 and buf\_x < 7:  
 if board\_checkers[buf\_y - 1][buf\_x + 1] == 0:  
 move.append([buf\_y - 1, buf\_x + 1])  
 else:  
 break  
 buf\_y -= 1  
 buf\_x += 1  
 else:  
 break  
 elif board\_checkers[buf\_y - 1][buf\_x + 1] == 0:  
 move.append([buf\_y - 1, buf\_x + 1])  
 else:  
 if board\_checkers[buf\_y - 1][buf\_x + 1] == 0:  
 move.append([buf\_y - 1, buf\_x + 1])  
 buf\_y -= 1  
 buf\_x += 1  
 kills[k].append(kill)  
 moves[k].append(move)  
 kill = []  
 move = []  
 buf\_x = x  
 buf\_y = y  
 while buf\_y < 7 and buf\_x < 7:  
 if board\_checkers[buf\_y + 1][buf\_x + 1] == p1 or board\_checkers[buf\_y + 1][buf\_x + 1] == p\_queen1:  
 break  
 elif buf\_y < 6 and buf\_x < 6:  
 if board\_checkers[buf\_y + 1][buf\_x + 1] == p2 or board\_checkers[buf\_y + 1][buf\_x + 1] == p\_queen2:  
 if board\_checkers[buf\_y + 2][buf\_x + 2] == 0:  
 move = []  
 kill.append([buf\_y + 1, buf\_x + 1])  
 move.append([buf\_y + 2, buf\_x + 2])  
 buf\_y += 1  
 buf\_x += 1  
 while buf\_y < 6 and buf\_x < 6:  
 if board\_checkers[buf\_y + 1][buf\_x + 1] == 0:  
 move.append([buf\_y + 1, buf\_x + 1])  
 else:  
 break  
 buf\_y += 1  
 buf\_x += 1  
 if board\_checkers[buf\_y + 1][buf\_x + 1] == 0:  
 move.append([buf\_y + 1, buf\_x + 1])  
 else:  
 break  
 elif board\_checkers[buf\_y + 1][buf\_x + 1] == 0:  
 move.append([buf\_y + 1, buf\_x + 1])  
 else:  
 if board\_checkers[buf\_y + 1][buf\_x + 1] == 0:  
 move.append([buf\_y + 1, buf\_x + 1])  
 buf\_y += 1  
 buf\_x += 1  
 kills[k].append(kill)  
 moves[k].append(move)  
 kill = []  
 move = []  
 buf\_x = x  
 buf\_y = y  
 while buf\_y < 7 and buf\_x > 0:  
 if board\_checkers[buf\_y + 1][buf\_x - 1] == p1 or board\_checkers[buf\_y + 1][buf\_x - 1] == p\_queen1:  
 break  
 elif buf\_y < 6 and buf\_x > 1:  
 if board\_checkers[buf\_y + 1][buf\_x - 1] == p2 or board\_checkers[buf\_y + 1][buf\_x - 1] == p\_queen2:  
 if board\_checkers[buf\_y + 2][buf\_x - 2] == 0:  
 move = []  
 kill.append([buf\_y + 1, buf\_x - 1])  
 move.append([buf\_y + 2, buf\_x - 2])  
 buf\_y += 1  
 buf\_x -= 1  
 while buf\_y < 6 and buf\_x > 1:  
 if board\_checkers[buf\_y + 1][buf\_x - 1] == 0:  
 move.append([buf\_y + 1, buf\_x - 1])  
 else:  
 break  
 buf\_y += 1  
 buf\_x -= 1  
 if board\_checkers[buf\_y + 1][buf\_x - 1] == 0:  
 move.append([buf\_y + 1, buf\_x - 1])  
 else:  
 break  
 elif board\_checkers[buf\_y + 1][buf\_x - 1] == 0:  
 move.append([buf\_y + 1, buf\_x - 1])  
 else:  
 if board\_checkers[buf\_y + 1][buf\_x - 1] == 0:  
 move.append([buf\_y + 1, buf\_x - 1])  
 buf\_y += 1  
 buf\_x -= 1  
 kills[k].append(kill)  
 moves[k].append(move)  
 return moves, kills, buf\_move  
  
  
def queen\_move(kills, moves, move):  
 global board\_checkers  
 global click1  
 global click2  
 global click\_stop  
 global move\_count  
 ind = []  
 p\_queen = player\_move + 10  
 for i in range(len(kills)):  
 for j in range(len(kills[i])):  
 if len(kills[i][j]) > 0:  
 ind.append([i, j])  
 if len(ind) > 0:  
 if click1 in move:  
 ind\_move = move.index(click1)  
 kills\_move = kills[ind\_move]  
 moves\_move = moves[ind\_move]  
 for i in range(4):  
 if len(kills\_move[i]) > 0:  
 if click2 in moves\_move[i]:  
 ind = moves\_move[i].index(click2)  
 board\_checkers[click1[0]][click1[1]] = 0  
 board\_checkers[kills\_move[i][0][0]][kills\_move[i][0][1]] = 0  
 board\_checkers[moves\_move[i][ind][0]][moves\_move[i][ind][1]] = p\_queen  
 board\_rendering()  
 moves, kills, move = queen\_check()  
 ind = []  
 for i1 in range(len(kills)):  
 for j in range(len(kills[i1])):  
 if len(kills[i1][j]) > 0:  
 ind.append([i1, j])  
 board\_rendering()  
 if len(ind) > 0:  
 click\_stop = False  
 click1 = click2  
 click2 = [0, 0]  
 else:  
 click\_stop = True  
 move\_count += 1  
 lbl\_count.configure(text=f"Осталось ходов {2 - move\_count}")  
 lbl\_info.configure(text="Ход выполнен")  
 click1 = [0, 0]  
 click2 = [0, 0]  
 else:  
 lbl\_info.configure(text="Нужно съесть вражескую шашку")  
 else:  
 flag = False  
 kill1, kill2, kill3 = check\_kill(player\_move)  
 if len(kill1) == 0:  
 for i in range(len(moves)):  
 for j in range(4):  
 if click2 in moves[i][j]:  
 board\_checkers[click1[0]][click1[1]] = 0  
 board\_checkers[click2[0]][click2[1]] = p\_queen  
 board\_rendering()  
 move\_count += 1  
 lbl\_info.configure(text="Ход выполнен")  
 click1 = [0, 0]  
 click2 = [0, 0]  
 flag = True  
 break  
 if flag:  
 break  
 else:  
 lbl\_info.configure(text="Нужно съесть вражескую шашку")  
 board\_rendering()  
  
  
def moved(kill1, kill2, kill3):  
 global board\_checkers  
 global move\_count  
 global click\_stop  
 global click1  
 global click2  
 ind = -1  
 if click1 in kill1:  
 c = kill2.count(click2)  
 for i in range(c+1):  
 ind = kill1.index(click1, ind + 1)  
 if ind == kill2.index(click2) or kill2[ind] == click2:  
 board\_checkers[click1[0]][click1[1]] = 0  
 board\_checkers[click2[0]][click2[1]] = player\_move  
 checker\_x = kill3[ind][1]  
 checker\_y = kill3[ind][0]  
 board\_checkers[checker\_y][checker\_x] = 0  
 board\_rendering()  
 kill1, kill2, kill3 = check\_kill(player\_move)  
 if click2 in kill1:  
 click\_stop = False  
 click1 = click2  
 click2 = [0, 0]  
 break  
 else:  
 click\_stop = True  
 move\_count += 1  
 lbl\_count.configure(text=f"Осталось ходов {2 - move\_count}")  
 lbl\_info.configure(text="Ход выполнен")  
 click1 = [0, 0]  
 click2 = [0, 0]  
 break  
 else:  
 lbl\_info.configure(text="Нужно съесть вражескую шашку")  
 board\_rendering()  
 become\_queen()  
  
  
def one\_move():  
 global click1  
 global click2  
 global board\_checkers  
 global move\_count  
 moves, kills, move = queen\_check()  
 ind = []  
 for i in range(len(kills)):  
 for j in range(len(kills[i])):  
 if len(kills[i][j]) > 0:  
 ind.append([i, j])  
 if len(ind) == 0:  
 if click2[0] - click1[0] == 1 and player\_move == 2:  
 board\_checkers[click1[0]][click1[1]] = 0  
 board\_checkers[click2[0]][click2[1]] = player\_move  
 move\_count += 1  
 lbl\_count.configure(text=f"Осталось ходов {2 - move\_count}")  
 lbl\_info.configure(text="Ход выполнен")  
 click1 = [0, 0]  
 click2 = [0, 0]  
 elif click2[0] - click1[0] == -1 and player\_move == 3:  
 board\_checkers[click1[0]][click1[1]] = 0  
 board\_checkers[click2[0]][click2[1]] = player\_move  
 move\_count += 1  
 lbl\_count.configure(text=f"Осталось ходов {2 - move\_count}")  
 lbl\_info.configure(text="Ход выполнен")  
 click1 = [0, 0]  
 click2 = [0, 0]  
 else:  
 lbl\_info.configure(text="Нужно съесть вражескую шашку")  
 become\_queen()  
 board\_rendering()  
  
  
def become\_queen():  
 global board\_checkers  
 for j in range(1, 8, 2):  
 if board\_checkers[0][j] == 3:  
 board\_checkers[0][j] = 13  
 for j in range(0, 7, 2):  
 if board\_checkers[7][j] == 2:  
 board\_checkers[7][j] = 12  
 board\_rendering()  
  
  
def check\_kill(p):  
 kill1 = []  
 kill2 = []  
 kill3 = []  
 if p == 3:  
 for y in range(8):  
 for x in range(8):  
 if board\_checkers[y][x] == 3:  
 if y > 1:  
 if x >= 6 or x <= 1:  
 if x >= 6:  
 if board\_checkers[y - 1][x - 1] == 2 or board\_checkers[y - 1][x - 1] == 12:  
 if board\_checkers[y - 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x - 2])  
 kill3.append([y - 1, x - 1])  
 else:  
 if board\_checkers[y - 1][x + 1] == 2 or board\_checkers[y - 1][x + 1] == 12:  
 if board\_checkers[y - 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x + 2])  
 kill3.append([y - 1, x + 1])  
 else:  
 if board\_checkers[y - 1][x - 1] == 2 or board\_checkers[y - 1][x - 1] == 12:  
 if board\_checkers[y - 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x - 2])  
 kill3.append([y - 1, x - 1])  
 if board\_checkers[y - 1][x + 1] == 2 or board\_checkers[y - 1][x + 1] == 12:  
 if board\_checkers[y - 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x + 2])  
 kill3.append([y - 1, x + 1])  
 if y < 6:  
 if x >= 6 or x <= 1:  
 if x >= 6:  
 if board\_checkers[y + 1][x - 1] == 2 or board\_checkers[y + 1][x - 1] == 12:  
 if board\_checkers[y + 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x - 2])  
 kill3.append([y + 1, x - 1])  
 else:  
 if board\_checkers[y + 1][x + 1] == 2 or board\_checkers[y + 1][x + 1] == 12:  
 if board\_checkers[y + 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x + 2])  
 kill3.append([y + 1, x + 1])  
 else:  
 if board\_checkers[y + 1][x - 1] == 2 or board\_checkers[y + 1][x - 1] == 12:  
 if board\_checkers[y + 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x - 2])  
 kill3.append([y + 1, x - 1])  
 if board\_checkers[y + 1][x + 1] == 2 or board\_checkers[y + 1][x + 1] == 12:  
 if board\_checkers[y + 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x + 2])  
 kill3.append([y + 1, x + 1])  
 else:  
 for y in range(8):  
 for x in range(8):  
 if board\_checkers[y][x] == 2:  
 if y < 6:  
 if x >= 6 or x <= 1:  
 if x > 5:  
 if board\_checkers[y + 1][x - 1] == 3 or board\_checkers[y + 1][x - 1] == 13:  
 if board\_checkers[y + 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x - 2])  
 kill3.append([y + 1, x - 1])  
 else:  
 if board\_checkers[y + 1][x + 1] == 3 or board\_checkers[y + 1][x + 1] == 13:  
 if board\_checkers[y + 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x + 2])  
 kill3.append([y + 1, x + 1])  
 else:  
 if board\_checkers[y + 1][x - 1] == 3 or board\_checkers[y + 1][x - 1] == 13:  
 if board\_checkers[y + 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x - 2])  
 kill3.append([y + 1, x - 1])  
 if board\_checkers[y + 1][x + 1] == 3 or board\_checkers[y + 1][x + 1] == 13:  
 if board\_checkers[y + 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y + 2, x + 2])  
 kill3.append([y + 1, x + 1])  
 if y > 1:  
 if x >= 6 or x <= 1:  
 if x > 5:  
 if board\_checkers[y - 1][x - 1] == 3 or board\_checkers[y - 1][x - 1] == 13:  
 if board\_checkers[y - 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x - 2])  
 kill3.append([y - 1, x - 1])  
 else:  
 if board\_checkers[y - 1][x + 1] == 3 or board\_checkers[y - 1][x - 1] == 13:  
 if board\_checkers[y - 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x + 2])  
 kill3.append([y - 1, x + 1])  
 else:  
 if board\_checkers[y - 1][x - 1] == 3 or board\_checkers[y - 1][x - 1] == 13:  
 if board\_checkers[y - 2][x - 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x - 2])  
 kill3.append([y - 1, x - 1])  
 if board\_checkers[y - 1][x + 1] == 3 or board\_checkers[y - 1][x - 1] == 13:  
 if board\_checkers[y - 2][x + 2] == 0:  
 kill1.append([y, x])  
 kill2.append([y - 2, x + 2])  
 kill3.append([y - 1, x + 1])  
 return kill1, kill2, kill3  
  
  
def minimax\_one\_move():  
 global click1  
 global click2  
 move\_click1 = []  
 move\_click2 = []  
 count = 0  
 for i in range(8):  
 for j in range(8):  
 if board\_checkers[i][j] == player\_move:  
 move\_click2.append([])  
 move\_click1.append([i, j])  
 if i + 1 < 8:  
 for j1 in range(8):  
 if abs(j - j1) == 1:  
 if board\_checkers[i + 1][j1] == 0 and canvases[i + 1][j1] == 1:  
 move\_click2[count].append([i + 1, j1])  
 count += 1  
 ind = []  
 for i in range(len(move\_click2)):  
 if move\_click2[i] == []:  
 ind.append(i)  
 ind.reverse()  
 for i in ind:  
 move\_click1.pop(i)  
 move\_click2.pop(i)  
 if move\_click1 != 0:  
 ind = randint(0, len(move\_click1) - 1)  
 click1 = move\_click1[ind]  
 click2 = move\_click2[ind][randint(0, len(move\_click2[ind]) - 1)]  
 one\_move()  
 board\_rendering()  
 return  
  
  
def minimax():  
 global player\_move  
 global move\_count  
 global click\_buf  
 global click1  
 global click2  
 if move\_count < 2:  
 moves, kills, move = queen\_check()  
 kill1, kill2, kill3 = check\_kill(player\_move)  
 ind = []  
 for j in range(len(kills)):  
 for i in range(len(kills[j])):  
 if kills[j][i] != []:  
 ind.append(i)  
 if len(ind) > 0:  
 i = randint(0, len(kills)-1)  
 click1 = move[i]  
 ind = []  
 for j in range(4):  
 if moves[i][j] == []:  
 ind.append(j)  
 ind.reverse()  
 for j in ind:  
 moves[i].pop(j)  
 buf1 = moves[i][randint(0, len(moves[i])-1)]  
 buf2 = buf1[randint(0, len(buf1) - 1)]  
 click2 = buf2  
 moves, kills, move = queen\_check()  
 queen\_move(kills, moves, move)  
 elif len(kill1) > 0 and move\_count < 2:  
 i = randint(0, len(kill1)-1)  
 click1 = kill1[i]  
 click2 = kill2[i]  
 killed\_checker(kill1, kill2, kill3)  
 elif move\_count < 2:  
 if randint(0, 1) == 0:  
 if len(moves) != 0:  
 i = randint(0, len(move) - 1)  
 click1 = move[i]  
 ind = []  
 for j in range(4):  
 if moves[i][j] == []:  
 ind.append(j)  
 ind.reverse()  
 for j in ind:  
 moves[i].pop(j)  
 buf1 = moves[i][randint(0, len(moves[i]) - 1)]  
 buf2 = buf1[randint(0, len(buf1) - 1)]  
 click2 = buf2  
 queen\_move(kills, moves, move)  
 else:  
 minimax\_one\_move()  
 board\_rendering()  
 if move\_count == 2:  
 move\_count = 0  
 player\_move = 3  
 lbl\_player.configure(text="Ход белых")  
 lbl\_count.configure(text=f"Осталось ходов {2 - move\_count}")  
 return  
 minimax()  
 board\_rendering()  
 check\_win()  
  
  
def check\_win():  
 c1 = 0  
 c2 = 0  
 for i in range(8):  
 c1 += board\_checkers[i].count(2) + board\_checkers[i].count(12)  
 c2 += board\_checkers[i].count(3) + board\_checkers[i].count(13)  
 if c1 == 0:  
 lbl\_text = "Победили красные"  
 elif c2 == 0:  
 lbl\_text = "Победили белые"  
 if c1 == 0 or c2 == 0:  
 info = Toplevel()  
 info.title("Конец игры")  
 info.geometry("200x200+100+100")  
 info.resizable(width=False, height=False)  
 info.grab\_set()  
 lbl = Label(info, text=lbl\_text, width=18, font="20")  
 btn\_ok = Button(info, text="Ок", width=5, command=lambda window2=info: close\_game(window2))  
 btn\_continue = Button(info, text="Ёще раз", width=10, command=lambda window2=info: game\_restart(window2))  
 lbl.grid(column=0, row=0, pady=20)  
 btn\_ok.grid(column=0, row=2, pady=15)  
 btn\_continue.grid(column=0, row=3)  
  
  
def close\_game(info):  
 global game\_window  
 info.destroy()  
 game\_window.destroy()  
  
  
def game\_restart(info):  
 global game\_window  
 global board\_checkers\_start  
 global board\_checkers  
 global move\_count  
 global player\_move  
 info.destroy()  
 game\_window.destroy()  
 move\_count = 0  
 player\_move = 3  
 board\_checkers = copy.deepcopy(board\_checkers\_start)  
 board()  
  
  
login\_fun()