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 «Царские башни»

Р.02069337. 22/2369 42 ТЗ-1

Листов: 47

Руководитель разработки:

к. т. н., доцент

Шишкин Вадим Викторинович

«29» декабря 2023 г.

Исполнитель:

студент гр. ИСТбд-22

Яхин Равиль Рустамович

«29» декабря 2023 г.

2023 г.

Содержание

Аннотация……...………………………………………………………….3

Техническое задание………………………………………………….......4

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

Руководство программиста……………………………….………….......17

Текст программы…..………………………………………………….......23

**Аннотация**

Данный документ представляет собой пояснительную записку на курсовую работу на тему «Царские башни». 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 средства.

**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 «Царские башни»

Р.02069337. 22/2369-42 ТЗ-1

Листов: 4

**Исполнитель**:

студент гр. ИСТбд-22

Яхин Равиль Рустамович

«29» декабря 2023 г.

2023 г.

**Введение**

Курсовая работа представляет собой десктопное приложение по теме игры царские башни.

Краткие правила игры:

В шашках царские башни работаю все те же правила, что и в русских шашках, но с некоторыми отличиями:

* В распоряжении игроков 11 простых шашек и одна шашка-царь.
* Выигрыш в царских башнях возможен тремя способами: либо путём захвата царя соперника, либо проведением своего царя на одну из четырёх ячеек превращения противоположной горизонтали, либо блокированием (лишение ходов) всех фишек соперника (как в русских шашках).

Функциональные возможности:

* Графический интерфейс взаимодействия с пользователем.
* Регистрация/авторизация пользователя.
* Проверка правильности и отрисовка ходов пользователя и компьютера.

**1. Основания для разработки**

Основанием для разработки является 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 царские шашки с графическим интерфейсом в среде Windows.

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

Приложение должно состоять из модуля (блока), который выполняет определенные функции по организации пользовательского интерфейса и самого процесса игры. При необходимости модуль должен обладать дополнительными информационными файлами.

2.2.2 Требования к составу функций приложения

В приложении должны быть реализованы в графическом режиме следующие

основные функции:

- регистрация/авторизация пользователя;

- реализация игрового поля;

- взаимодействие с пользователем;

- проверка правильности и отрисовка хода пользователя;

- проверка окончания игры;

- вычисление, проверка правильности и отрисовка хода компьютера;

- сообщение об окончании игры и победителе.

2.2.3 Требования к организации информационного обеспечения, входных и выходных данных

В десктопном приложении должен быть реализован графический интерфейс для взаимодействия с пользователем. Окно с полями для ввода логина и пароля, которые вводятся с клавиатуры и хранятся в отдельном файле или базе данных в зашифрованном виде. Кнопка регистрации/авторизации пользователя. Вывод окна игрового поля – шашки. Изображения шашек могут храниться в отдельных файлах. Пояснительные информационные сообщения для пользователя должны выводиться по ходу игры.

**2.3 Требования к надёжности**

Программа должна нормально функционировать при бесперебойной работе ЭВМ. При возникновении сбоя в работе аппаратуры, восстановление нормальной работы программы должно производиться после: перезагрузки операционной системы; запуска исполняемого файла программы; повторного выполнения действий, потерянных до последнего сохранения информации в файл на диске. Уровень надёжности программы должен соответствовать технологии программирования, предусматривающей: инспекцию исходных текстов программы; автономное тестирование модулей (методов) программы; тестирование сопряжении модулей (методов) программы; комплексное тестирование программы.

**2.4 Требования к информационной и программной совместимости**

Операционная система: Windows 11

Используемые библиотеки: tkinter

Язык: Python 3.10.4

Среда разработки: Visual Studio Code

**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

Все файлы проекта должны хранится в специально отведённом репозитории онлайн-сервиса GitHub.

2.6 3 Сроки хранения

Срок хранения – до июля 2026 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 «Царские башни»

**Пояснительная записка**

Р.02069337. 22/2369-42 ПЗ-01

Листов: 7

**Исполнитель**:

студент гр. ИСТбд-22

Яхин Равиль   
Рустамович

«29» декабря 2023 г.

2023 г.

**Введение**

Курсовая работа представляет собой десктопное приложение по теме игры царские башни.

Краткое описание реализованного приложения:

* Графический интерфейс взаимодействия с пользователем.
* Регистрация/авторизация пользователя. Хеширование пароля.
* Проверка правильности и отрисовка ходов пользователя и компьютера.
* Оценка и выбор наилучшего хода.
* Определение победителя и возможность переигровки.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу. Детализируется в разработанном техническом задании.

**1.2 Математические методы**

Модель:

pole=[[0,3,0,6,0,3,0,3],

[3,0,3,0,3,0,3,0],

[0,3,0,3,0,3,0,3],

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

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

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

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

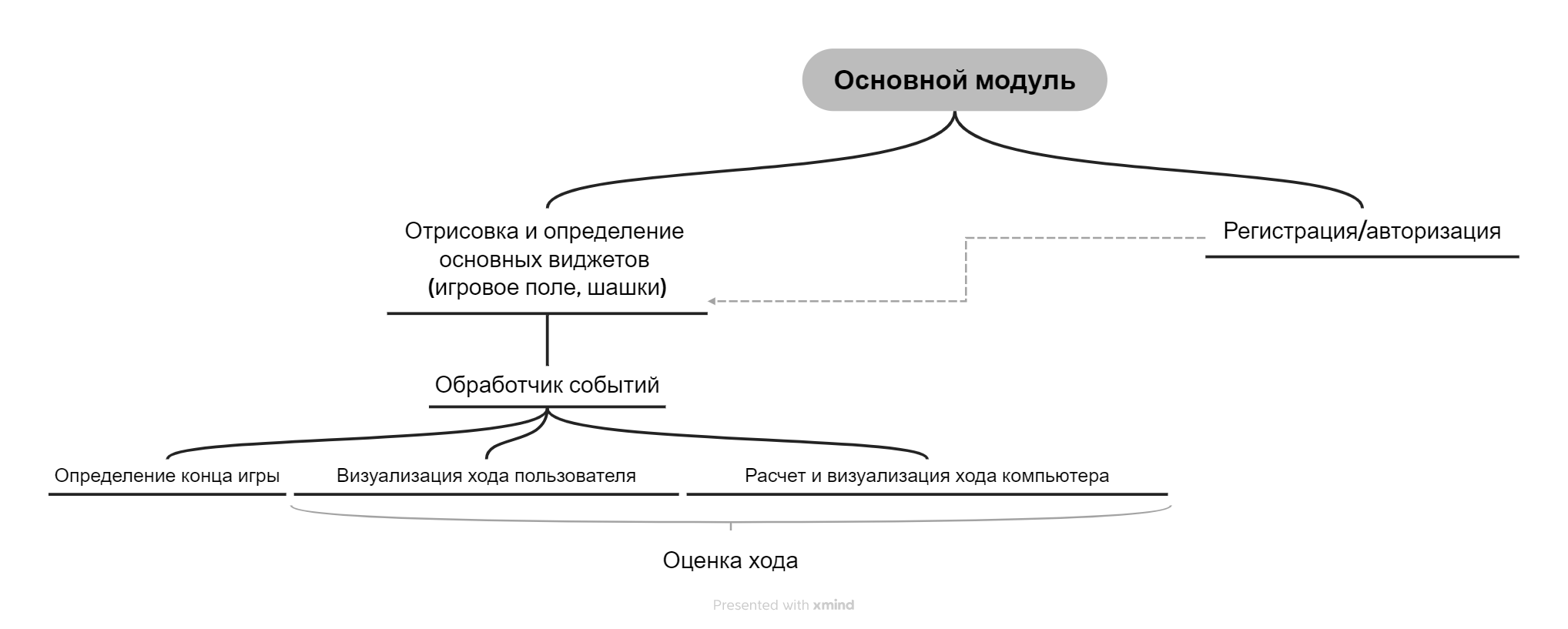
[1,0,1,0,5,0,1,0]]

1 – белые шашки, 3 – черные шашки, 0 – свободные клетки, 5 – белая шашка-царь, 6 – черная шашка-царь.

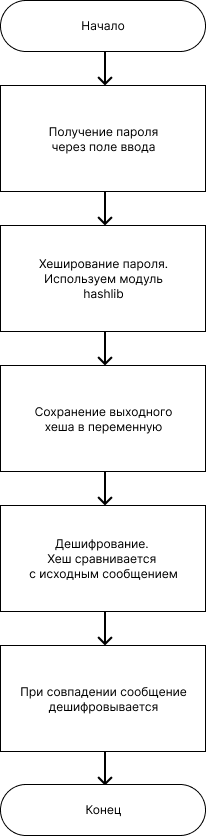
Дамки белых шашек будут обозначаться цифрой 2, черных – 4. В программе реализован анализ позиции шашки: если вычисляется, что шашка находиться на противоположной горизонтали (крайней), то происходит превращение в дамку.

**1.3 Архитектура и алгоритмы**

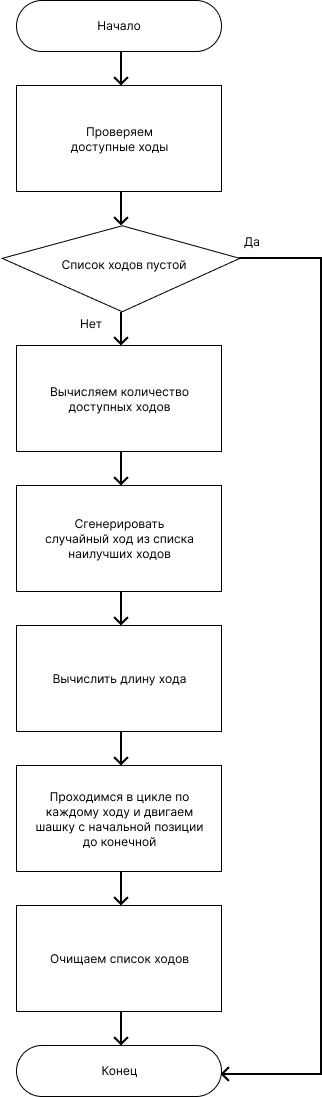
1.3.1. Архитектура



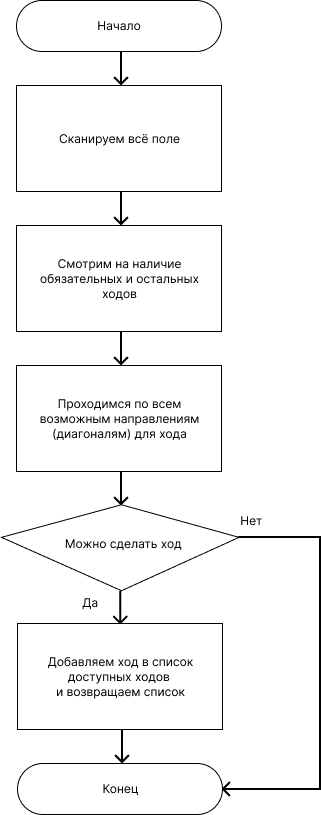
1.3.2. Алгоритм шифрования и дешифрования



1.3.3. Алгоритм хода



1.3.4. Алгоритм проверки доступности хода



1.3.5. Алгоритм отрисовки поля

Данный алгоритм включает в себя два алгоритма, первый из которых рисует на поле сетку, а второй шашки.

1.3.6. Алгоритм входа и регистрации

Данный алгоритм включает в себя два алгоритма и предназначены для осуществления регистрации пользователя и последующего входа в главное окно игры.

**1.4 Тестирование**

1.4.1 Описание отчета о тестировании

В данном отчете представлены результаты тестирования программы на основе функционального тестирования, тестирования удобства пользования, тестирования на отказ и восстановление. Описаны проведенные тесты, их результаты и обнаруженные дефекты.

1.4.2 Цель тестирования

Целью тестирования является проверка соответствия ПО предъявляемым требованиям, а также выявление возможных багов. По результатам тестирования следует исправление выявленных багов.

1.4.3 Методика тестирования

* Функциональное тестирование;
* Тестирование удобства пользования;
* Тестирование на отказ и восстановление.

1.4.4 Проведенные тесты

Отработка авторизации.

Предварительные шаги:

Зарегистрироваться с логином: user и паролем 123user.

Шаги:

1. Запустить приложение.

2. В окне регистрации, в поле «логин» ввести user, а в поле «пароль» - 123user.

3. Нажать кнопку «Войти».

Ожидаемый результат: Пользователь начнет игру.

Фактический результат: Пользователь начал новую игру.

Симулировать внезапный отказ электричества на компьютере (обесточить компьютер).

1.4.5 Выводы

На основе проведенных тестов сделаны следующие выводы:

– Программа успешно прошла все тесты и работает корректно.

– Рекомендации по дальнейшему улучшению программы: добавление звукового сопровождения, таблицы лидеров.

**2. Источники, использованные при разработке**

1. Введение в Tkinter // Habr URL: https://habr.com/ru/post/133337/ (дата обращения: 26.10.2023).

2. "Крестики-нолики" с алгоритмом "Минимакс" URL: https://www.youtube.com/watch?v=JoJI10CFLzI (дата обращения: 18.11.2023).

3. Tkinter — создание графического интерфейса в Python // python-scripts URL: https://python-scripts.com/tkinter (дата обращения: 02.12.2023).

4. Python Checkers AI Tutorial Part 2 — Реализация и визуализация (Minimax) // YouTube URL: Python Checkers AI Tutorial Part 2 — Реализация и визуализация (Minimax) – YouTube (дата обращения: 16.11.2023).

5. Путь к пониманию декораторов в Python // Habr URL: https://habr.com/ru/company/wunderfund/blog/657355/ (дата обращения: 20.11.2023).

6. Checkers-Python // Medium URL: https://medium.com/analytics-vidhya/checkers-python-eff2786b985b (дата обращения: 21.11.2023)

7. Python Checkers// YouTube URL:  
https://youtube.com/playlist?list=PLzMcBGfZo4lkJr3sqpikNyVzbNZLRiT3&si=UkWBXgBuCdS6SnAo (дата обращения 13.12.2023)

**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 «Царские башни»

**Руководство программиста**

Р.02069337. 22/2369-42 РП-01

Листов: 6

Исполнитель:

студент гр. ИСТбд-22

Яхин Равиль   
Рустамович

«29» декабря 2023 г.

2023 г.

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Десктопное приложение по теме игры Шашки царские башни.

Краткие правила игры:

В шашках царские башни работаю все те же правила, что и в русских шашках, но с некоторыми отличиями:

* В распоряжении игроков 11 простых шашек и одна шашка-царь.
* Выигрыш в царских башнях возможен тремя способами: либо путём захвата царя соперника, либо проведением своего царя на одну из четырёх ячеек превращения противоположной горизонтали, либо блокированием (лишение ходов) всех фишек соперника (как в русских шашках).

Функциональные возможности:

* Графический интерфейс взаимодействия с пользователем.
* Регистрация/авторизация пользователя.
* Проверка правильности и отрисовка ходов пользователя и компьютера.

**1.2 Условия, необходимые для использования приложения**

Приложение можно использовать на персональном компьютере. Для использования приложения необходимы:

1. OC Windows 7,8,10,11;
2. Язык Python версии 3.9.
3. Библиотеки: tkinter, copy, random, hashlib, time, os.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Количество значимых строк кода – 507.

Количество алгоритмов – 9.

Библиотеки tkinter, copy, random, hashlib, time, os

Порядок работы:

После запуска на экране монитора появится окно авторизации (рис. 1), на котором есть кнопки «Войти» и «Зарегистрироваться».

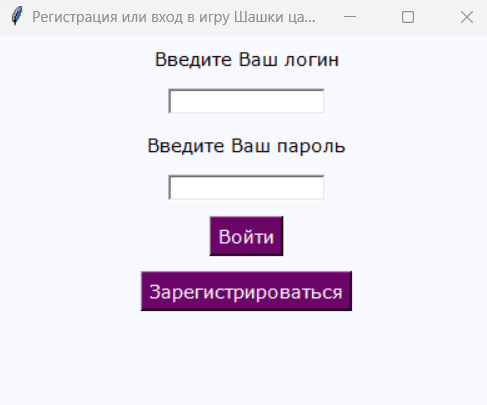


Рис. 1. Окно авторизации.

После введения данных и нажатия на кнопку «Зарегистрироваться» появляется окно с текстом об успешной регистрации аккаунта и просьбой заново войти в свой аккаунт.

При успешной авторизации открывается окно игры. (Рис. 2)

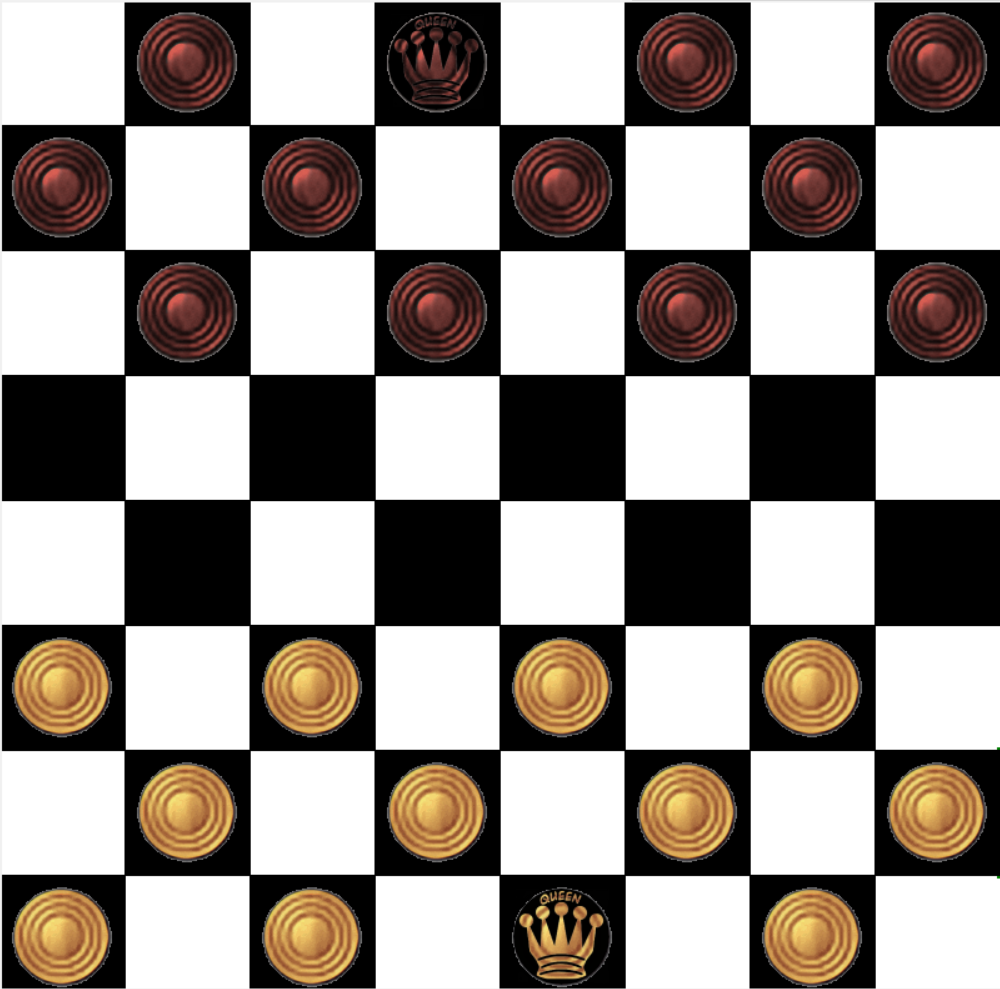


Рис. 2. Окно игры.

Далее пользователю следует левой кнопкой мыши выбрать шашку, которой он хочет пойти, и далее указать соседнюю клетку с ней для хода (рис. 3).

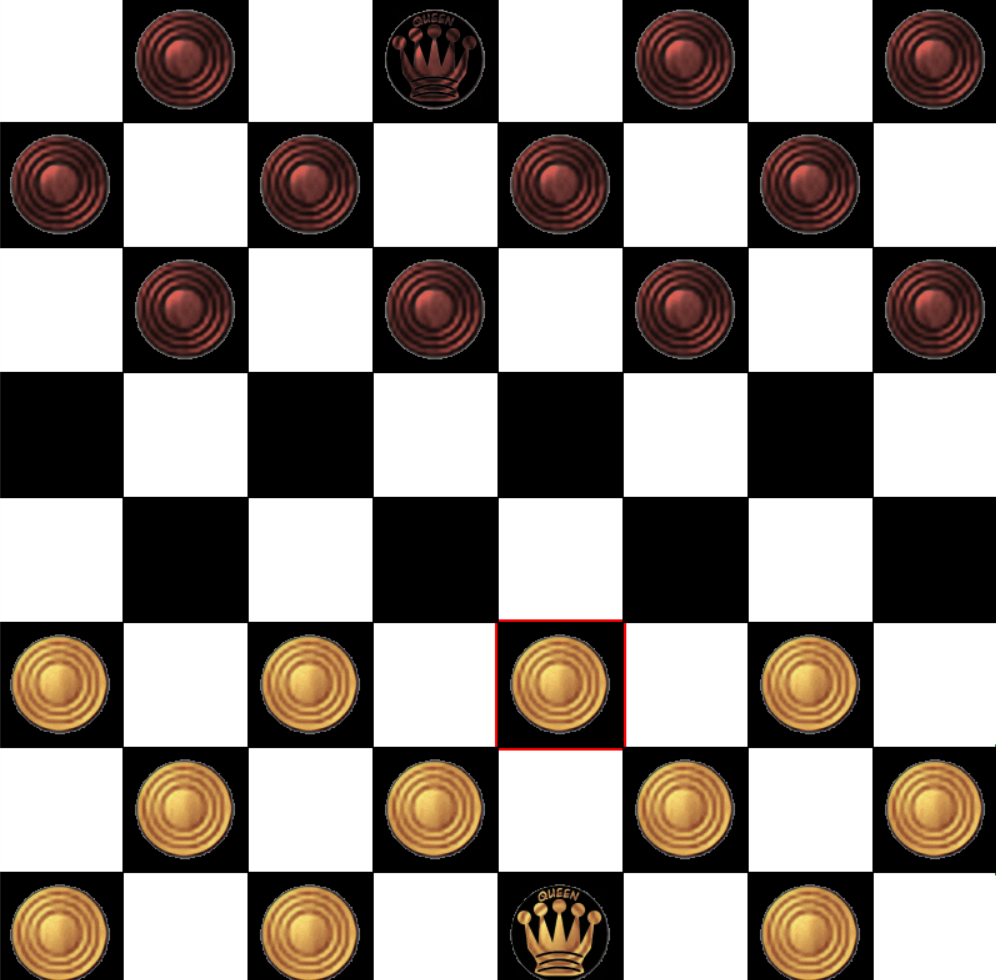


Рис. 3. Шашка выбрана игроком и помечена красной рамкой.

После хода белыми шашками, право хода приходит черным, ход будет делать компьютер. (Рис. 4)

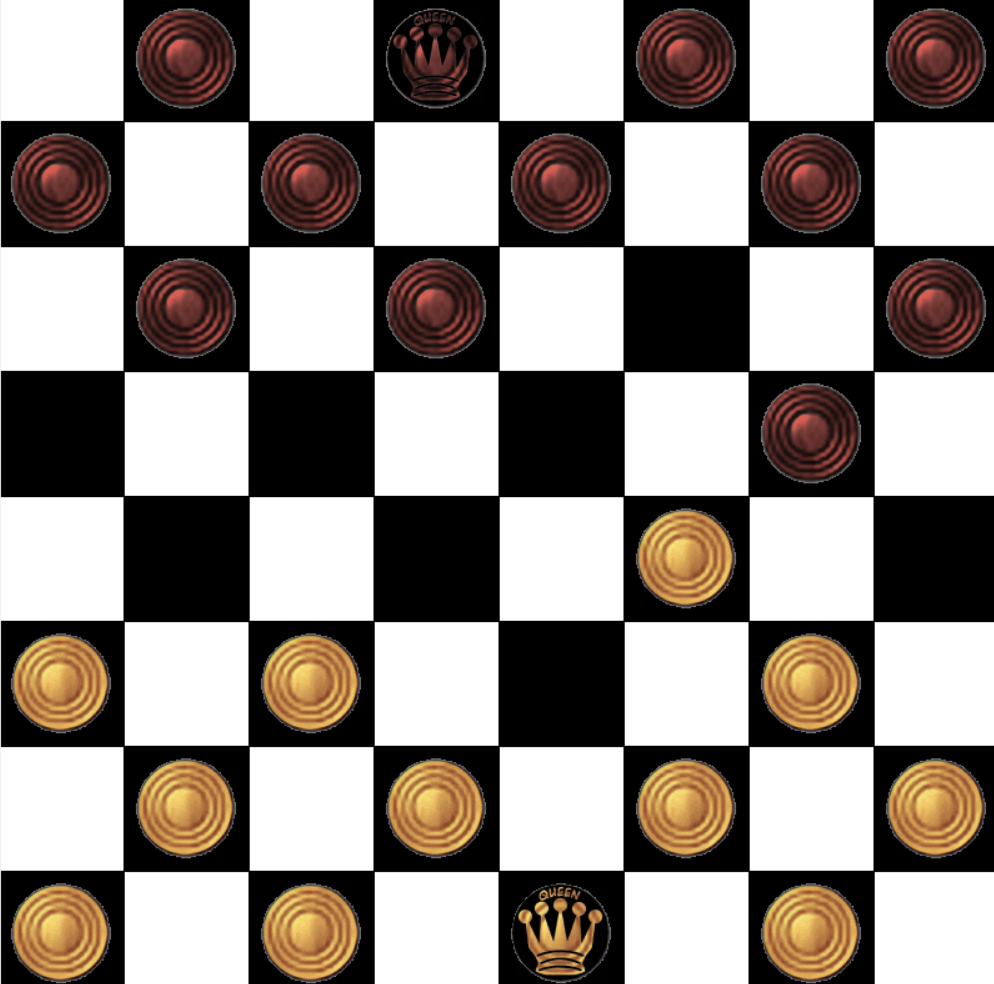


Рис. 4. Шашка компьютера сделала ход.

После того как у кого-то из игроков закончились шашки, либо кто-то съест короля соперника, либо чей-то король перейдет на противоположную крайнюю горизонталь, программа выдаст сообщение о победе соответствующего игрока. (Рис. 5)

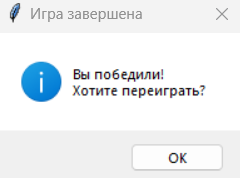


Рис. 5. Игра завершена.

При нажатии на кнопку «ОК» пользователю предоставляется возможность сыграть еще партию.

**2.2 Особенности реализации приложения**

В программе используются массивы, отвечающие за координаты игрового поля (расстановку шашек игрока и противника), наличие ходов, нахождение шашек.

**3. Обращение к программе**

Алгоритмы:

1. «hash\_password» - алгоритм, отвечающий за хеширование пароля пользователя.
2. «enter\_users» - алгоритм, отвечающий за вход пользователя в игру
3. «draw\_board» – алгоритм, отвечающий за отрисовку игрового поля и шашек на ней.
4. «igrok\_move» - алгоритм, обрабатывающий ход игрока.
5. «сomputers\_move» - алгоритм, обрабатывающий ход компьютера.
6. «spisok\_hodov\_igroka» - алгоритм, составляющий список ходов игрока
7. «spisok\_hodov\_komp» - алгоритм, составляющий список ходов компьютера
8. «check\_computer\_movement» - алгоритм, проверяющий наличие ходов компьютера и подбирающий наилучший ход
9. «checkers\_counting» - алгоритм, подсчитывающий количество шашек на поле

Библиотеки:

1. tkinter – для работы с графическим интерфейсом игры
2. copy - для копирования игрового поля
3. random – для выбора рандомного хода из доступных
4. hashlib – для хеширования пароля пользователей
5. time – для задержки хода компьютера
6. os – для работы с файлом данных о зарегистрированных пользователях

**4. Сообщения**

При окончании игры программа отображает окно, в виде сообщений: «Вы победили! Хотите переиграть?» или «Вы проиграли! Хотите переиграть?» или «Ходов больше нет. Хотите переиграть?».

При вводе неправильного логина или пароля или при пустых окнах ввода при попытке входа всплывет окно с сообщением «Неверный логин или пароль».

При вводе неправильного логина или пароля или при пустых окнах ввода при попытке зарегистрироваться всплывет окно с сообщением «Поля Логин и Пароль должны быть заполнены».

При вводе повторных данных при регистрации выведется окно «Учетная запись с таким логином уже существует».

При успешной регистрации откроется окно с сообщением «Регистрация успешно завершена. При следующем входе введите свои данные, чтобы войти в игру!».

При успешном входе откроется окно с сообщением «Вы вошли в свой аккаунт».

**Текст программы:**

from tkinter import \*

import random

import time

import copy

import os

from tkinter import messagebox

from tkinter import font

from tkinter.messagebox import showinfo, showerror

import hashlib

spisok\_hodov=()#конечный список ходов компьютера

predict\_hodov=3#сколько ходов компьютер предсказывает

k\_rezult=0

o\_rezult=0

poz1\_x=-1#клетка не определена

f\_hodigroka=True#определение хода игрока, по умолчанию активен игрок

def hash\_password(password): # Хеширование пароля

return hashlib.sha256(password.encode('utf-8')).hexdigest()

def check\_login(): # Проверяем наличие логина в файле

if os.path.exists("users.txt"):

file = open("users.txt", "r+")

with open("users.txt", "r") as file:

lines = file.readlines()

login\_input = login.get()

for line in lines:

if login\_input in line:

return True

return False

def check\_users(): # Проверяем наличие данных в файле о пользователе

if os.path.exists("users.txt"):

file = open("users.txt", "r+")

lines = file.readlines()

login\_input = login.get()

password\_input = password.get()

for line in lines:

parts = line.strip().split(':')

if len(parts) == 2:

stored\_login, stored\_password = parts

if login\_input == stored\_login and hash\_password(password\_input) == stored\_password:

return True

return False

def registration\_user(): # Регистрируем пользователя

if not login.get() or not password.get():

showerror("Ошибка", "Поля 'Логин' и 'Пароль' должны быть заполнены.")

elif check\_login():

showerror("Ошибка", "Учетная запись с таким логином уже существует.")

else:

with open("users.txt", "a") as file:

file.write(f"{login.get()}:{hash\_password(password.get())}\n")

root.destroy()

showinfo("Успех", "Регистрация успешно завершена.\nПри следующем входе введите свои данные, чтобы войти в игру!")

def enter\_users():

if check\_users(): #если такой пользователь существует

showinfo("Успех!", "Вы вошли в свой аккаунт")

main\_window=Toplevel()

main\_window.title('Шашки царские башни')

doska=Canvas(main\_window, width=800,height=800,bg='#FFFFFF')

doska.pack()

def checkers\_images():#загружаем шашки

global checkers

i1=PhotoImage(file="checkers\_images\\white.gif")

i2=PhotoImage(file="checkers\_images\\white\_king.gif")

i3=PhotoImage(file="checkers\_images\\black.gif")

i4=PhotoImage(file="checkers\_images\\black\_king.gif")

i5=PhotoImage(file="checkers\_images\\qween\_shashki.png")

i6=PhotoImage(file="checkers\_images\\qween\_shashkib.png")

checkers=[0,i1,i2,i3,i4,i5,i6]

def new\_game():#начинаем новую партию игры

global pole

pole=[[0,3,0,6,0,3,0,3],

[3,0,3,0,3,0,3,0],

[0,3,0,3,0,3,0,3],

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

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

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

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

[1,0,1,0,5,0,1,0]]

def draw\_board(x\_poz\_1,y\_poz\_1,x\_poz\_2,y\_poz\_2):#рисуем шашечную доску

global checkers

global pole

global red\_outline,green\_outline

k=100

x=0

doska.delete('all')

red\_outline=doska.create\_rectangle(-5, -5, -5, -5,outline="red",width=5)

green\_outline=doska.create\_rectangle(-5, -5, -5, -5,outline="green",width=5)

while x<8\*k:

y=1\*k

while y<8\*k:

doska.create\_rectangle(x, y, x+k, y+k,fill="#292424")

y+=2\*k

x+=2\*k

x=1\*k

while x<8\*k:

y=0

while y<8\*k:

doska.create\_rectangle(x, y, x+k, y+k,fill="#292424")

y+=2\*k

x+=2\*k

for y in range(8):#распологаем шашки

for x in range(8):

z=pole[y][x] #проходимся по строкам [y] и столбцам [x]

if z:

if (x\_poz\_1,y\_poz\_1)!=(x,y):#стоячие шашки

doska.create\_image(x\*k,y\*k, anchor=NW, image=checkers[z])

#рисуем активную шашку

z=pole[y\_poz\_1][x\_poz\_1]

if z: #если позиция не равна 0, т. е. есть шашка

doska.create\_image(x\_poz\_1\*k,y\_poz\_1\*k, anchor=NW, image=checkers[z],tag='ani')

#анимация шашек

kx = 1 if x\_poz\_1<x\_poz\_2 else -1

ky = 1 if y\_poz\_1<y\_poz\_2 else -1

for i in range(abs(x\_poz\_1-x\_poz\_2)):#анимация движения шашки

for ii in range(33):

doska.move('ani',0.03\*k\*kx,0.03\*k\*ky)

doska.update()

time.sleep(0.01)

def message\_output(mes): # выводим сообщение об окончании игры

global f\_hodigroka

title='Игра завершена'

if mes==1:

i=messagebox.showinfo(title=title, message='Вы победили!\nХотите переиграть?',icon='info')

if mes==2:

i=messagebox.showinfo(title=title, message='Вы проиграли!\nХотите переиграть?',icon='info')

if mes==3:

i=messagebox.showinfo(title=title, message='Ходов больше нет.\nХотите переиграть?',icon='info')

if i:

new\_game()

draw\_board(-1,-1,-1,-1)#рисуем игровое поле

f\_hodigroka=True #игрок может делать ход

def position\_one(event):#определяем клетку для хода 1

x,y=(event.x)//100,(event.y)//100#вычисляем координаты клетки

doska.coords(green\_outline,x\*100,y\*100,x\*100+100,y\*100+100)#рамка в выбранной клетке

def position\_two(event):#определяем клетку для хода 2

global poz1\_x,poz1\_y,poz2\_x,poz2\_y

global f\_hodigroka

x,y=(event.x)//100,(event.y)//100#определяем координаты клетки

if pole[y][x]==1 or pole[y][x]==2 or pole[y][x]==5:#если шашка игрока в выбранной клетке

doska.coords(red\_outline,x\*100,y\*100,x\*100+100,y\*100+100)#рамка в выбранной клетке

poz1\_x,poz1\_y=x,y

else:

if poz1\_x!=-1:#клетка выбрана

poz2\_x,poz2\_y=x,y

if f\_hodigroka:#если ход игрока

igrok\_move()

if not(f\_hodigroka):#если ход компьютера

time.sleep(0.5)

computers\_move()#передаём ход компьютеру

poz1\_x=-1#клетка не определена

doska.coords(red\_outline,-5,-5,-5,-5)#рамка вне поля

def computers\_move():

global f\_hodigroka

global spisok\_hodov

check\_computer\_movement(1,(),[])

if spisok\_hodov:#проверяем наличие доступных ходов

kol\_hodov=len(spisok\_hodov)#вычисляем количество доступных ходов в списке `spisok\_hodov` и сохраняем его в переменной

tek\_hod=random.randint(0,kol\_hodov-1)#генерируем случайное целое число в диапазоне от 0 до `kol\_hodov-1` и сохраняем его в переменной. Это число будет использоваться для выбора случайного хода из списка `spisok\_hodov`

deep\_hod=len(spisok\_hodov[tek\_hod])#вычисляем длину выбранного случайного хода из списка `spisok\_hodov`

for i in range(deep\_hod-1):

#выполняем ход

move\_checkers(1,spisok\_hodov[tek\_hod][i][0],spisok\_hodov[tek\_hod][i][1],spisok\_hodov[tek\_hod][1+i][0],spisok\_hodov[tek\_hod][1+i][1])

spisok\_hodov=[]#очищаем список ходов

f\_hodigroka=True#игрок может делать ход

s\_komp\_checkers,s\_igrok\_checkers,s\_iqween,s\_kqween=checkers\_counting()

if not(s\_igrok\_checkers) or not(s\_iqween) or check\_victory\_komputer():

message\_output(2)

elif not(s\_komp\_checkers) or not(s\_kqween) or check\_victory\_igrok():

message\_output(1)

elif f\_hodigroka and not(spisok\_hodov\_igroka()):

message\_output(3)

elif not(f\_hodigroka) and not(spisok\_hodov\_komp()):

message\_output(3)

def spisok\_hodov\_komp():#генерируем список ходов компьютера

spisok=check\_hod\_komp1([])#проверяем обязательные ходы

if not(spisok):

spisok=check\_hod\_komp2([])#проверяем оставшиеся ходы

return spisok

def check\_computer\_movement(tek\_hod,n\_spisok,spisok):#!!!

global pole

global spisok\_hodov

global best\_rezult,k\_rezult,o\_rezult

if not(spisok):#если список ходов пустой...

spisok=spisok\_hodov\_komp()#заполняем

if spisok:

k\_pole=copy.deepcopy(pole)

for ((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)) in spisok:#проходим по всем ходам из списка

tek\_spisok\_hod=move\_checkers(0,poz1\_x,poz1\_y,poz2\_x,poz2\_y)

if tek\_spisok\_hod:#если существует ещё ход

check\_computer\_movement(tek\_hod,(n\_spisok+((poz1\_x,poz1\_y),)),tek\_spisok\_hod)

else:

check\_igrok\_movement(tek\_hod,[])

if tek\_hod==1:

t\_rez=o\_rezult/k\_rezult

if not(spisok\_hodov):#записываем если пустой

spisok\_hodov=(n\_spisok+((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)),)

best\_rezult=t\_rez#записываем лучший результат

else:

if t\_rez==best\_rezult:

spisok\_hodov=spisok\_hodov+(n\_spisok+((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)),)

if t\_rez>best\_rezult:

spisok\_hodov=()

spisok\_hodov=(n\_spisok+((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)),)

best\_rezult=t\_rez#записываем лучший результат

o\_rezult=0

k\_rezult=0

pole=copy.deepcopy(k\_pole)#возвращаем поле

else:

s\_komp\_checkers,s\_igrok\_checkers,s\_iqween,s\_kqween=checkers\_counting()#подсчёт результата хода

o\_rezult+=(s\_komp\_checkers+s\_kqween-s\_igrok\_checkers-s\_iqween)

k\_rezult+=1

def checkers\_counting():#подсчёт шашек на поле

global pole

s\_igrok\_checkers=0

s\_iqween = 0

s\_komp\_checkers=0

s\_kqween = 0

for i in range(8):

for ii in pole[i]:

if ii==1:

s\_igrok\_checkers+=1

if ii==5:

s\_iqween+=1

if ii==2:

s\_igrok\_checkers+=3

if ii==3:

s\_komp\_checkers+=1

if ii==6:

s\_kqween+=1

if ii==4:

s\_komp\_checkers+=3

return s\_komp\_checkers,s\_igrok\_checkers,s\_iqween,s\_kqween

def check\_victory\_igrok():

global pole

player\_queen\_position = None # позиция царской шашки игрока

# Поиск позиций царских шашек игрока и врага

for i in range(8):

for j in range(8):

if pole[i][j] == 5:

player\_queen\_position = (i, j)

# Проверка, достигла ли царская шашка игрока противоположной горизонтали на одну из четырех клеток

if player\_queen\_position[0] == 0:

return True

return False

def check\_victory\_komputer():

global pole

enemy\_queen\_position = None # позиция царской шашки врага

# Поиск позиций царских шашек игрока и врага

for i in range(8):

for j in range(8):

if pole[i][j] == 6:

enemy\_queen\_position = (i, j)

# Проверка, достигла ли царская шашка врага противоположной горизонтали на одну из четырех клеток

if enemy\_queen\_position is not None and enemy\_queen\_position[0] == 7:

return True

return False

def igrok\_move():

global poz1\_x,poz1\_y,poz2\_x,poz2\_y

global f\_hodigroka

f\_hodigroka=False#считаем ход игрока выполненным

spisok=spisok\_hodov\_igroka()

if spisok:

if ((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)) in spisok:#проверяем ход на соблюдение правил игры

tek\_spisok\_hod=move\_checkers(1,poz1\_x,poz1\_y,poz2\_x,poz2\_y)#если всё хорошо, делаем перемещаем шашку

if tek\_spisok\_hod:#если есть ещё ход той же шашкой

f\_hodigroka=True#считаем ход игрока невыполненным

else:

f\_hodigroka=True#считаем ход игрока невыполненным

doska.update()

s\_komp\_checkers,s\_igrok\_checkers,s\_iqween,s\_kqween=checkers\_counting()

if not(s\_igrok\_checkers) or not(s\_iqween) or check\_victory\_komputer():

message\_output(2)

elif not(s\_komp\_checkers) or not(s\_kqween) or check\_victory\_igrok():

message\_output(1)

elif f\_hodigroka and not(spisok\_hodov\_igroka()):

message\_output(3)

elif not(f\_hodigroka) and not(spisok\_hodov\_komp()):

message\_output(3)

def spisok\_hodov\_igroka():#составляем список ходов игрока

spisok=check\_hod\_igrok1([])#проверяем обязательные ходы

if not(spisok):

spisok=check\_hod\_igrok2([])#проверяем оставшиеся ходы

return spisok

def check\_hod\_igrok1(spisok):#проверка наличия обязательных ходов

spisok=[]#список ходов

for y in range(8):#сканируем всё поле

for x in range(8):

spisok=check\_hod\_igrok1p(spisok,x,y)

return spisok

def check\_hod\_igrok1p(spisok,x,y):

if pole[y][x]==1 or pole[y][x]==5:#шашки

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

if 0<=y+iy+iy<=7 and 0<=x+ix+ix<=7:

if pole[y+iy][x+ix]==3 or pole[y+iy][x+ix]==4 or pole[y+iy][x+ix]==6:

if pole[y+iy+iy][x+ix+ix]==0:

spisok.append(((x,y),(x+ix+ix,y+iy+iy)))#запись хода в конец списка

if pole[y][x]==2:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

true\_move=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if true\_move==1:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6:

true\_move+=1 # увеличиваем на 1, если на пути встречается вражеская фигура

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5 or true\_move==2: #если на пути встречается своя фигура, то...

if true\_move>0:spisok.pop()#...удаляем ход из списка

break

return spisok

def check\_hod\_igrok2(spisok):#проверка наличия остальных ходов

for y in range(8):#сканируем всё поле

for x in range(8):

if pole[y][x]==1 or pole[y][x]==5:#шашки

for ix,iy in (-1,-1),(1,-1):

if 0<=y+iy<=7 and 0<=x+ix<=7:

if pole[y+iy][x+ix]==0:

spisok.append(((x,y),(x+ix,y+iy)))#запись хода в конец списка

if pole[y+iy][x+ix]==3 or pole[y+iy][x+ix]==4 or pole[y+iy][x+ix]==6:

if 0<=y+iy\*2<=7 and 0<=x+ix\*2<=7:

if pole[y+iy\*2][x+ix\*2]==0:

spisok.append(((x,y),(x+ix\*2,y+iy\*2)))#запись хода в конец списка

if pole[y][x]==2:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

true\_move=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if pole[y+iy\*i][x+ix\*i]==0:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6:

true\_move+=1

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5 or true\_move==2:

break

return spisok

def check\_igrok\_movement(tek\_hod,spisok):

global pole,k\_rezult,o\_rezult

global predict\_hodov

if not(spisok):

spisok=spisok\_hodov\_igroka()

if spisok:#проверяем наличие доступных ходов

k\_pole=copy.deepcopy(pole)#копируем поле

for ((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)) in spisok: # перебираем каждый возможный ход в списке

tek\_spisok\_hod=move\_checkers(0,poz1\_x,poz1\_y,poz2\_x,poz2\_y)

if tek\_spisok\_hod:#если существует ещё ход,...

check\_igrok\_movement(tek\_hod,tek\_spisok\_hod) #... то рекурсивно проверяем остальные ходы

else: # если ходов больше нет

if tek\_hod<predict\_hodov: # если текущий ход меньше максимального количества ходов predict\_hodov

check\_computer\_movement(tek\_hod+1,(),[])

else:

s\_komp\_checkers,s\_igrok\_checkers,s\_kqween,s\_iqween=checkers\_counting()#подсчёт результата хода

o\_rezult+=(s\_komp\_checkers+s\_kqween-s\_igrok\_checkers-s\_iqween)

k\_rezult+=1

pole=copy.deepcopy(k\_pole)#возвращаем поле

else:#доступных ходов нет

s\_komp\_checkers,s\_igrok\_checkers,s\_kqween,s\_iqween=checkers\_counting()#подсчёт результата хода

o\_rezult+=(s\_komp\_checkers+s\_kqween-s\_igrok\_checkers-s\_iqween)

k\_rezult+=1

def move\_checkers(f,poz1\_x,poz1\_y,poz2\_x,poz2\_y):

global pole

if f:draw\_board(poz1\_x,poz1\_y,poz2\_x,poz2\_y)#рисуем игровое поле

#превращение

if poz2\_y==0 and pole[poz1\_y][poz1\_x]==1:

pole[poz1\_y][poz1\_x]=2

if poz2\_y==7 and pole[poz1\_y][poz1\_x]==3:

pole[poz1\_y][poz1\_x]=4

pole[poz2\_y][poz2\_x]=pole[poz1\_y][poz1\_x]

pole[poz1\_y][poz1\_x]=0

#уничтожаем шашку игрока

kx=ky=1

if poz1\_x<poz2\_x: # если координаты x позиции начальной шашки меньше координаты x позиции конечной шашки

kx=-1

if poz1\_y<poz2\_y: # если координаты y позиции начальной шашки меньше координаты y позиции конечной шашки

ky=-1

x\_poz,y\_poz=poz2\_x,poz2\_y # переменные x\_poz и y\_poz устанавливаются равными координатам конечной позиции шашки

while (poz1\_x!=x\_poz) or (poz1\_y!=y\_poz): # цикл будет выполняться до тех пор, пока координаты начальной и конечной позиции шашки не станут равными

x\_poz+=kx

y\_poz+=ky

# перебираем все клетки на пути атакующей шашки от начальной до конечной позиции

if pole[y\_poz][x\_poz]!=0: # если текущая клетка на игровом поле не пуста, то она считается занятой шашкой противника...

pole[y\_poz][x\_poz]=0 #...эта шашка удаляется из игры

if f:draw\_board(-1,-1,-1,-1)#рисуем игровое поле

#проверяем ход той же шашкой...

if pole[poz2\_y][poz2\_x]==3 or pole[poz2\_y][poz2\_x]==4 or pole[poz2\_y][poz2\_x]==6:#...компьютера

return check\_hod\_komp1p([],poz2\_x,poz2\_y)#возвращаем список доступных ходов

elif pole[poz2\_y][poz2\_x]==1 or pole[poz2\_y][poz2\_x]==2 or pole[poz2\_y][poz2\_x]==5:#...игрока

return check\_hod\_igrok1p([],poz2\_x,poz2\_y)#возвращаем список доступных ходов

if f:draw\_board(poz1\_x,poz1\_y,poz2\_x,poz2\_y)#рисуем игровое поле

def check\_hod\_komp1(spisok):#проверка наличия обязательных ходов, возвращаем список возможных ходов для всех шашек компьютера.

for y in range(8):#сканируем всё поле

for x in range(8):

spisok=check\_hod\_komp1p(spisok,x,y)

return spisok

def check\_hod\_komp1p(spisok,x,y):

if pole[y][x]==3 or pole[y][x]==6:#шашки

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

if 0<=y+iy+iy<=7 and 0<=x+ix+ix<=7:

if pole[y+iy][x+ix]==1 or pole[y+iy][x+ix]==2 or pole[y+iy][x+ix]==5: # проверяем возможность "перепрыгнуть" через вражескую шашку

if pole[y+iy+iy][x+ix+ix]==0: # понимаем, что есть возможность сделать ход, так как клетка равна 0

spisok.append(((x,y),(x+ix+ix,y+iy+iy)))#запись хода в конец списка

if pole[y][x]==4:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

true\_move=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if true\_move==1:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5:

true\_move+=1

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6 or true\_move==2:

if true\_move>0:spisok.pop()#удаление хода из списка

break

return spisok

def check\_hod\_komp2(spisok):#проверка наличия остальных ходов

for y in range(8):#сканируем всё поле

for x in range(8):

if pole[y][x]==3 or pole[y][x]==6:#шашки

for ix,iy in (-1,1),(1,1):

if 0<=y+iy<=7 and 0<=x+ix<=7:

if pole[y+iy][x+ix]==0:

spisok.append(((x,y),(x+ix,y+iy)))#запись хода в конец списка

if pole[y+iy][x+ix]==1 or pole[y+iy][x+ix]==2 or pole[y+iy][x+ix]==5:

if 0<=y+iy\*2<=7 and 0<=x+ix\*2<=7:

if pole[y+iy\*2][x+ix\*2]==0:

spisok.append(((x,y),(x+ix\*2,y+iy\*2)))#запись хода в конец списка

if pole[y][x]==4:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

true\_move=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if pole[y+iy\*i][x+ix\*i]==0:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5:

true\_move+=1

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6 or true\_move==2:

break

return spisok

checkers\_images()#загружаем шашки

new\_game()##начинаем новую партию игры

draw\_board(-1,-1,-1,-1)#рисуем шашечную доску

doska.bind("<Motion>", position\_one)

doska.bind("<Button-1>", position\_two)

root.withdraw()

main\_window.protocol("WM\_DELETE\_WINDOW", root.destroy)

mainloop()

else:

showerror("Ошибка", "Неверный логин или пароль.")

root = Tk() # Создаем окно

root.title("Регистрация или вход в игру Шашки царские башни") # Устанавливаем заголовок окна

root.geometry("400x300") # Устанавливаем размеры окна

root.configure(background="#F8F8FF")

font1 = font.Font(family="Verdana", size=11, weight="normal", slant="roman")

llogin = Label(font=font1, anchor=W, background="#F8F8FF", text="Введите Ваш логин")

llogin.pack(padx=6, pady=6)

login = Entry(bd=2)

login.pack(padx=6, pady=6)

lpassword = Label(font=font1, anchor=W, background="#F8F8FF", text="Введите Ваш пароль")

lpassword.pack(padx=6, pady=6)

password = Entry(bd=2, show="\*")

password.pack(padx=6, pady=6)

btn1 = Button(text="Войти", bg="#6b0568", fg="#FFFFFF", font=font1, command=enter\_users)

btn1.pack(padx=6, pady=6)

btn2 = Button(text="Зарегистрироваться", bg="#6b0568", fg="#FFFFFF", font=font1, command=registration\_user)

btn2.pack(padx=6, pady=6)

root.mainloop()

def skan():#подсчёт шашек на поле

global pole

s\_i=0

s\_iqween = 0

s\_k=0

s\_kqween = 0

for i in range(8):

for ii in pole[i]:

if ii==1:s\_i+=1

if ii==5:s\_iqween+=1

if ii==2:s\_i+=3

if ii==3:s\_k+=1

if ii==6:s\_kqween+=1

if ii==4:s\_k+=3

return s\_k,s\_i,s\_iqween,s\_kqween

def check\_victory\_igrok():

global pole

player\_queen\_position = None # позиция царской шашки игрока

# Поиск позиций царских шашек игрока и врага

for i in range(8):

for j in range(8):

if pole[i][j] == 5:

player\_queen\_position = (i, j)

# Проверка, достигла ли царская шашка игрока противоположной горизонтали на одну из четырех клеток

if player\_queen\_position[0] == 0:

return True

return False

def check\_victory\_komputer():

global pole

enemy\_queen\_position = None # позиция царской шашки врага

# Поиск позиций царских шашек игрока и врага

for i in range(8):

for j in range(8):

if pole[i][j] == 6:

enemy\_queen\_position = (i, j)

# Проверка, достигла ли царская шашка врага противоположной горизонтали на одну из четырех клеток

if enemy\_queen\_position is not None and enemy\_queen\_position[0] == 7:

return True

return False

def hod\_igroka():

global poz1\_x,poz1\_y,poz2\_x,poz2\_y

global f\_hi

f\_hi=False#считаем ход игрока выполненным

spisok=spisok\_hi()

if spisok:

if ((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)) in spisok:#проверяем ход на соответствие правилам игры

t\_spisok=hod(1,poz1\_x,poz1\_y,poz2\_x,poz2\_y)#если всё хорошо, делаем ход

if t\_spisok:#если есть ещё ход той же пешкой

f\_hi=True#считаем ход игрока невыполненным

else:

f\_hi=True#считаем ход игрока невыполненным

doska.update()#!!!обновление

s\_k,s\_i,s\_iqween,s\_kqween=skan()

if not(s\_i) or not(s\_iqween) or check\_victory\_komputer():

soobsenie(2)

elif not(s\_k) or not(s\_kqween) or check\_victory\_igrok():

soobsenie(1)

elif f\_hi and not(spisok\_hi()):

soobsenie(3)

elif not(f\_hi) and not(spisok\_hk()):

soobsenie(3)

def spisok\_hi():#составляем список ходов игрока

spisok=prosmotr\_hodov\_i1([])#здесь проверяем обязательные ходы

if not(spisok):

spisok=prosmotr\_hodov\_i2([])#здесь проверяем оставшиеся ходы

return spisok

def prosmotr\_hodov\_i1(spisok):#проверка наличия обязательных ходов

spisok=[]#список ходов

for y in range(8):#сканируем всё поле

for x in range(8):

spisok=prosmotr\_hodov\_i1p(spisok,x,y)

return spisok

def prosmotr\_hodov\_i1p(spisok,x,y):

if pole[y][x]==1 or pole[y][x]==5:#шашки

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

if 0<=y+iy+iy<=7 and 0<=x+ix+ix<=7:

if pole[y+iy][x+ix]==3 or pole[y+iy][x+ix]==4 or pole[y+iy][x+ix]==6:

if pole[y+iy+iy][x+ix+ix]==0:

spisok.append(((x,y),(x+ix+ix,y+iy+iy)))#запись хода в конец списка

if pole[y][x]==2:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

osh=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if osh==1:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6:

osh+=1 # увеличиваем на 1, если на пути встречается вражеская фигура

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5 or osh==2: #если на пути встречается своя фигура, то...

if osh>0:spisok.pop()#...удаляем ход из списка

break

return spisok

def prosmotr\_hodov\_i2(spisok):#проверка наличия остальных ходов

for y in range(8):#сканируем всё поле

for x in range(8):

if pole[y][x]==1 or pole[y][x]==5:#шашки

for ix,iy in (-1,-1),(1,-1):

if 0<=y+iy<=7 and 0<=x+ix<=7:

if pole[y+iy][x+ix]==0:

spisok.append(((x,y),(x+ix,y+iy)))#запись хода в конец списка

if pole[y+iy][x+ix]==3 or pole[y+iy][x+ix]==4 or pole[y+iy][x+ix]==6:

if 0<=y+iy\*2<=7 and 0<=x+ix\*2<=7:

if pole[y+iy\*2][x+ix\*2]==0:

spisok.append(((x,y),(x+ix\*2,y+iy\*2)))#запись хода в конец списка

if pole[y][x]==2:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

osh=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if pole[y+iy\*i][x+ix\*i]==0:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6:

osh+=1

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5 or osh==2:

break

return spisok

def proverka\_hi(tur,spisok):

global pole,k\_rez,o\_rez

global ur

if not(spisok):

spisok=spisok\_hi()

if spisok:#проверяем наличие доступных ходов

k\_pole=copy.deepcopy(pole)#копируем поле

for ((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)) in spisok: # перебираем каждый возможный ход в списке

t\_spisok=hod(0,poz1\_x,poz1\_y,poz2\_x,poz2\_y)

if t\_spisok:#если существует ещё ход,...

proverka\_hi(tur,t\_spisok) #... то рекурсивно вызываем функцию для проверки оставшихся ходов

else: # если ходов больше нет

if tur<ur: # если текущий ход меньше максимального количества ходов ur

proverka\_hk(tur+1,(),[])

else:

s\_k,s\_i,s\_kqween,s\_iqween=skan()#подсчёт результата хода

o\_rez+=(s\_k+s\_kqween-s\_i-s\_iqween)

k\_rez+=1

pole=copy.deepcopy(k\_pole)#возвращаем поле

else:#доступных ходов нет

s\_k,s\_i,s\_kqween,s\_iqween=skan()#подсчёт результата хода

o\_rez+=(s\_k+s\_kqween-s\_i-s\_iqween)

k\_rez+=1

def hod(f,poz1\_x,poz1\_y,poz2\_x,poz2\_y):

global pole

if f:vivod(poz1\_x,poz1\_y,poz2\_x,poz2\_y)#рисуем игровое поле

#превращение

if poz2\_y==0 and pole[poz1\_y][poz1\_x]==1:

pole[poz1\_y][poz1\_x]=2

if poz2\_y==7 and pole[poz1\_y][poz1\_x]==3:

pole[poz1\_y][poz1\_x]=4

pole[poz2\_y][poz2\_x]=pole[poz1\_y][poz1\_x]

pole[poz1\_y][poz1\_x]=0

#рубим шашку игрока

kx=ky=1

if poz1\_x<poz2\_x: # если координаты x позиции начальной шашки меньше координаты x позиции конечной шашки

kx=-1

if poz1\_y<poz2\_y: # если координаты y позиции начальной шашки меньше координаты y позиции конечной шашки

ky=-1

x\_poz,y\_poz=poz2\_x,poz2\_y # переменные x\_poz и y\_poz устанавливаются равными координатам конечной позиции шашки

while (poz1\_x!=x\_poz) or (poz1\_y!=y\_poz): # цикл будет выполняться до тех пор, пока координаты начальной и конечной позиции шашки не станут равными

x\_poz+=kx

y\_poz+=ky

# Это позволяет перебирать все клетки на пути атакующей шашки от начальной до конечной позиции

if pole[y\_poz][x\_poz]!=0: # если текущая клетка на игровом поле не пуста, то она считается занятой шашкой противника...

pole[y\_poz][x\_poz]=0 #...эта шашка удаляется из игры

if f:vivod(-1,-1,-1,-1)#рисуем игровое поле

#проверяем ход той же шашкой...

if pole[poz2\_y][poz2\_x]==3 or pole[poz2\_y][poz2\_x]==4 or pole[poz2\_y][poz2\_x]==6:#...компьютера

return prosmotr\_hodov\_k1p([],poz2\_x,poz2\_y)#возвращаем список доступных ходов

elif pole[poz2\_y][poz2\_x]==1 or pole[poz2\_y][poz2\_x]==2 or pole[poz2\_y][poz2\_x]==5:#...игрока

return prosmotr\_hodov\_i1p([],poz2\_x,poz2\_y)#возвращаем список доступных ходов

if f:vivod(poz1\_x,poz1\_y,poz2\_x,poz2\_y)#рисуем игровое поле

def prosmotr\_hodov\_k1(spisok):#проверка наличия обязательных ходов # возвращает список возможных ходов для всех шашек компьютера.

for y in range(8):#сканируем всё поле

for x in range(8):

spisok=prosmotr\_hodov\_k1p(spisok,x,y)

return spisok

def prosmotr\_hodov\_k1p(spisok,x,y):

if pole[y][x]==3 or pole[y][x]==6:#шашки

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

if 0<=y+iy+iy<=7 and 0<=x+ix+ix<=7:

if pole[y+iy][x+ix]==1 or pole[y+iy][x+ix]==2 or pole[y+iy][x+ix]==5: # проверяем возможность "перепрыгнуть" через вражескую шашку

if pole[y+iy+iy][x+ix+ix]==0: # понимаем, что есть возможность сделать ход, так как клетка равна 0

spisok.append(((x,y),(x+ix+ix,y+iy+iy)))#запись хода в конец списка

if pole[y][x]==4:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

osh=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if osh==1:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5:

osh+=1

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6 or osh==2:

if osh>0:spisok.pop()#удаление хода из списка

break

return spisok

def prosmotr\_hodov\_k2(spisok):#проверка наличия остальных ходов

for y in range(8):#сканируем всё поле

for x in range(8):

if pole[y][x]==3 or pole[y][x]==6:#шашки

for ix,iy in (-1,1),(1,1):

if 0<=y+iy<=7 and 0<=x+ix<=7:

if pole[y+iy][x+ix]==0:

spisok.append(((x,y),(x+ix,y+iy)))#запись хода в конец списка

if pole[y+iy][x+ix]==1 or pole[y+iy][x+ix]==2 or pole[y+iy][x+ix]==5:

if 0<=y+iy\*2<=7 and 0<=x+ix\*2<=7:

if pole[y+iy\*2][x+ix\*2]==0:

spisok.append(((x,y),(x+ix\*2,y+iy\*2)))#запись хода в конец списка

if pole[y][x]==4:#шашка с короной

for ix,iy in (-1,-1),(-1,1),(1,-1),(1,1):

osh=0#определение правильности хода

for i in range(1,8):

if 0<=y+iy\*i<=7 and 0<=x+ix\*i<=7:

if pole[y+iy\*i][x+ix\*i]==0:

spisok.append(((x,y),(x+ix\*i,y+iy\*i)))#запись хода в конец списка

if pole[y+iy\*i][x+ix\*i]==1 or pole[y+iy\*i][x+ix\*i]==2 or pole[y+iy\*i][x+ix\*i]==5:

osh+=1

if pole[y+iy\*i][x+ix\*i]==3 or pole[y+iy\*i][x+ix\*i]==4 or pole[y+iy\*i][x+ix\*i]==6 or osh==2:

break

return spisok

izobrazheniya\_peshek()#здесь загружаем изображения шашек

novaya\_igra()#начинаем новую игру

vivod(-1,-1,-1,-1)#рисуем игровое поле

doska.bind("<Motion>", pozici\_1)#движение мышки по полю

doska.bind("<Button-1>", pozici\_2)

root.withdraw()

gl\_okno.protocol("WM\_DELETE\_WINDOW", root.destroy)

mainloop()

else:

showerror("Ошибка", "Неверный логин или пароль.")

root = Tk() # Создаем окно

root.title("Регистрация или вход в игру Шашки царские башни") # Устанавливаем заголовок окна

root.geometry("400x300") # Устанавливаем размеры окна

root.configure(background="#F8F8FF")

font1 = font.Font(family="Verdana", size=11, weight="normal", slant="roman")

llogin = Label(font=font1, anchor=W, background="#F8F8FF", text="Введите Ваш логин")

llogin.pack(padx=6, pady=6)

login = Entry(bd=2)

login.pack(padx=6, pady=6)

lpassword = Label(font=font1, anchor=W, background="#F8F8FF", text="Введите Ваш пароль")

lpassword.pack(padx=6, pady=6)

password = Entry(bd=2, show="\*")

password.pack(padx=6, pady=6)

btn1 = Button(text="Войти", bg="#6b0568", fg="#FFFFFF", font=font1, command=enter\_users)

btn1.pack(padx=6, pady=6)

btn2 = Button(text="Зарегистрироваться", bg="#6b0568", fg="#FFFFFF", font=font1, command=registration\_user)

btn2.pack(padx=6, pady=6)

root.mainloop()