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»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Тема: Компьютерная логическая игра «Скифские шашки»

Р.02069337.22/2393-41ТЗ-01

Листов: 41

Руководитель разработки:

к. т. н., доцент

Шишкин Вадим Викторинович

«10» января 2024 г.

Исполнитель:

студент гр. ИСТбд-22

*Шиле Андрей Михайлович*

«10» января 2024 г.

2024 г.

Содержание

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

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

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

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

Текст программы…..………………………………………………….......24

**Аннотация**

Данный документ представляет собой пояснительную записку на курсовую работу на тему «Скифские шашки». 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»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Тема: Компьютерная логическая игра «Скифские шашки»

Р.02069337.22/2393-41ТЗ-01

Листов 5

**Исполнитель:**

студент гр. ИСТбд-22

*Шиле Андрей Михайлович*

«10» января 2024 г.

2024 г.

**Введение**

Компьютерная логическая игра «Скифские шашки». Игра ведется на доске 9х9 клеток. Шашки занимают первые три ряда с каждой стороны, располагаясь на зеленых и желтых клетках.

    Противники ходят поочередно, перемещая шашки своего цвета. **Обычная шашка** (в начале партии 13 шашек обычных и 1 дамка – вождь, который находиться в первом ряде по середине) может совершать два типа ходов.

* **тихий ход** - перемещение шашки на одно поле вперед по диагонали. Согласно правил игры, тихий ход возможен, если соответствующее поле свободно и нигде на доске не требуется выполнить взятие шашки соперника.
* **взятие шашки противника** - перемещение шашки на два поля вперед или назад по диагонали, перепрыгивая при этом через шашку противника, которая затем снимается с доски (**простая шашка может бить назад**).

    Когда обычная шашка достигает последней горизонтали, **она становится дамкой.** Дамки способны перемещаться на произвольное количество полей по диагонали, причем как вперед, так и назад.

    По правилам игры - взятия шашек противника обязательны. Если после взятия одной, есть возможным побить ещё одну шашку соперника, взятие продолжается. Если существует несколько вариантов битья, бить необходимо максимально возможное количество шашек.

**По правилам игры, выигрывает** тот, кому удалось уничтожить или заблокировать движение всех шашек противника.

**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 10

Используемые библиотеки: tkinter, os, time, copy, random, hashlib

Язык: Python 3.10.4

Среда разработки: PyCharm Comminity 2022.3.2.

**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 2024 года.

**3. Требования к программной документации**

Определяются заданием на курсовую работу.

**4. Стадии и этапы разработки**

Определяются заданием на курсовую работу.

**5. Порядок контроля и приёмки**

Определяются заданием на курсовую работу.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

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

Тема: Компьютерная логическая игра «Скифские шашки»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

**Пояснительная записка**

Р.02069337.22/2393-41ТЗ-01

Листов: 8

**Исполнитель**:

студент гр. ИСТбд-22

*Шиле Андрей Михайлович*

«10» января 2024 г.

2024 г.

**Введение**

Курсовая работа представляет собой десктопное приложение по теме игры скифские шашки.

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:

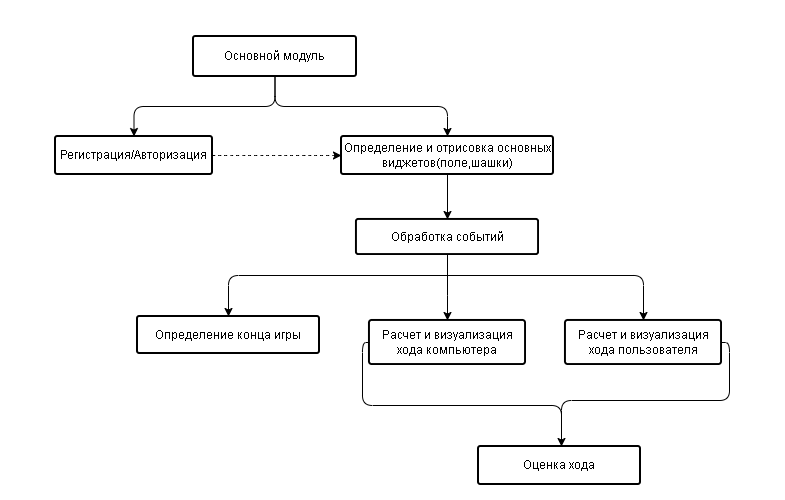
pole=[[3,0,3,0,4,0,3,0,3],  
 [0,3,0,3,0,3,0,3,0],  
 [3,0,3,0,3,0,3,0,3],  
 [0,0,0,0,0,0,0,0,0],  
 [0,0,0,0,0,0,0,0,0],  
 [0,0,0,0,0,0,0,0,0],  
 [1,0,1,0,1,0,1,0,1],  
 [0,1,0,1,0,1,0,1,0],  
 [1,0,1,0,2,0,1,0,1]]

0 – свободные клетки, 3 – черные шашки, 4 – черные дамки, 1 – белые шашки, 2 – белые дамки

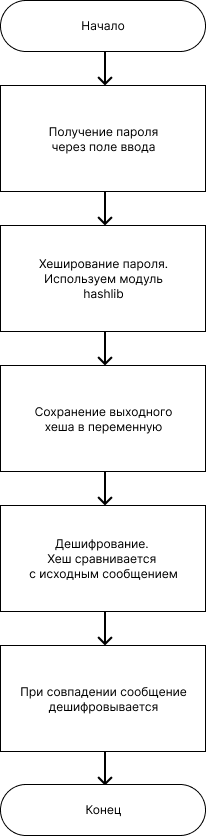
В программе реализован анализ позиции шашки: если вычисляется, что шашка находиться на противоположной горизонтали (крайней), то происходит превращение в дамку.

**1.3 Архитектура и алгоритмы**

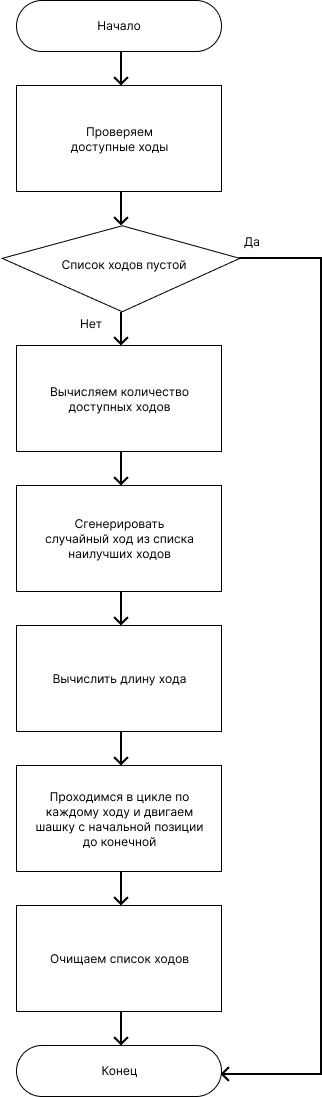
1.3.1. Архитектура



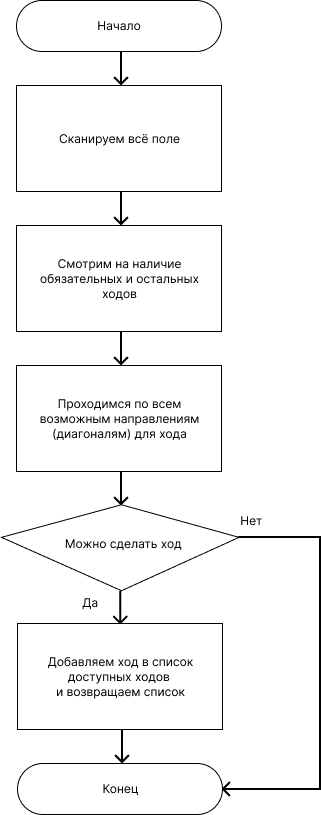
1.3.2. Алгоритм шифрования и дешифрования



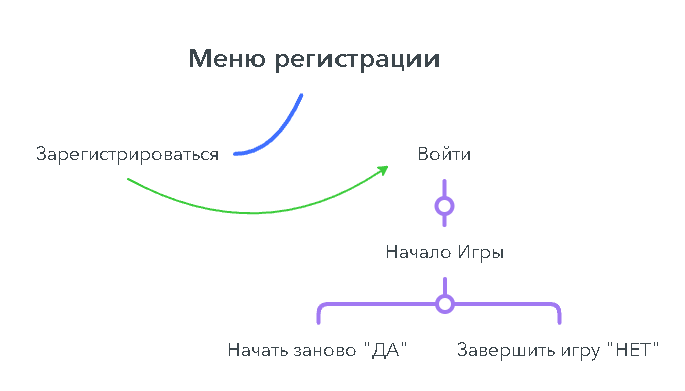
1.3.3 Алгоритм хода



1.3.4. Алгоритм проверки доступности хода



**1.4 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» - 123.

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 scripts [Электронный ресурс] Режим доступа: для всех пользователей.- URL: https://habr.com/ru/post/133337/ (дата обращения: 03.11.2023).

2. Tkinter — создание графического интерфейса в Python // python-scripts [Электронный ресурс] Режим доступа: для всех пользователей.- URL: https://python-scripts.com/tkinter (дата обращения: 02.12.2023).

3.Разработка логических компьютерных игр с графическим интерфейсом в среде Python [Электронный ресурс] / В.В. Шишкин, Д.С. Афонин. – Ульяновск: УлГТУ, 2023.– Режим доступа: для всех пользователей. – URL: <http://lib.ulstu.ru/venec/disk/2023/112.pdf> (дата обращения: 30.10.23).

4.Python Checkers// YouTube [Электронный ресурс] Режим доступа: для всех пользователей. - URL:

https://www.youtube.com/watch?v=vnd3RfeG3NM (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 «Скифские шашки»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

**Руководство программиста**

Р.02069337.22/2393-41ТЗ-01

Листов: 7

**Исполнитель**:

студент гр. ИСТбд-22

*Шиле Андрей Михайлович*

«10» января 2024 г.

2024 г.

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложение предназначено для развлечения, развития интеллекта, совершенствования способности игры в шахматы.

Правила игры:

Игра придерживается правил русских шашек;

Игра проводится на доске размера 9x9;

Игра начинается за белых(красных);

Ходы выполняются по очереди;

Игрок может передвигать только свои фигуры;

В начале игры у каждого по 14 шашек , 1 из которых – дамка;

Дамка ходит по диагонали на любое количество клеток;

Пешка двигается по диагонали;

Если достигнуть последнюю противоположную вашим шашкам горизонталь, то та шашка, которая на ней, становиться дамкой;

Если есть возможность бить вражескую шашку бьем;

Выигрывает тот, кто перебьет все вражеские шашки, или заблокирует;

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. Операционная система: Приложение должно быть доступно для Windows, macOS и Linux.

2. Платформа: Приложение должно быть написано на Python и совместимо с версиями Python 3 и выше.

3. Инструментальная среда: Для использования приложения необходимо установить Python и настроить его на компьютере пользователя.

4.Библиотеки: tkinter, copy, random, hashlib, time, os.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Количество значимых строк кода – 476

Количество алгоритмов – 9

Описание используемых библиотек.

1. Библиотека tkinter: это стандартная библиотека для создания графического интерфейса пользователя (GUI) в Python. Она предоставляет различные виджеты и методы для создания окон, кнопок, полей ввода, меню и других элементов интерфейса. Tkinter обеспечивает простой способ взаимодействия с пользователем и создания приятного пользовательского опыта.

2.Библиотека random: Модуль random предоставляет функции для генерации случайных чисел, букв, случайного выбора элементов последовательности.

3.Библиотека copy:Операция присваивания не копирует объект, он лишь создаёт ссылку на объект. Для изменяемых коллекций, или для коллекций, содержащих изменяемые элементы, часто необходима такая копия, чтобы её можно было изменить, не изменяя оригинал. Данный модуль предоставляет общие (поверхностная и глубокая) операции копирования.

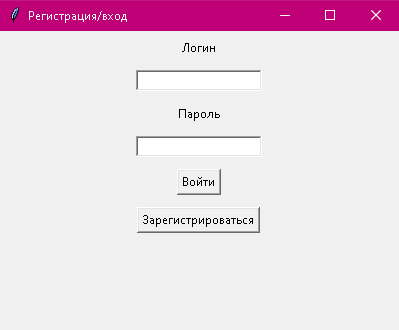
4. Библиотека hashlib: [Модуль hashlib](https://docs-python.ru/standart-library/modul-hashlib-python/) реализует общий интерфейс для множества различных безопасных алгоритмов хеширования и дайджеста сообщений. В модуль включены алгоритмы безопасного хеширования, такие как FIPS SHA1, SHA224, SHA256, SHA384 и SHA512, определенные в FIPS 180-2, а также алгоритм MDA RSA, определенный в Интернете RFC 1321.

5.Библиотека time: Time - модуль для работы со временем в Python.

6.Библиотека os: Модуль os предоставляет множество функций для работы с операционной системой, причём их поведение, как правило, не зависит от ОС, поэтому программы остаются переносимыми. Здесь будут приведены наиболее часто используемые из них.

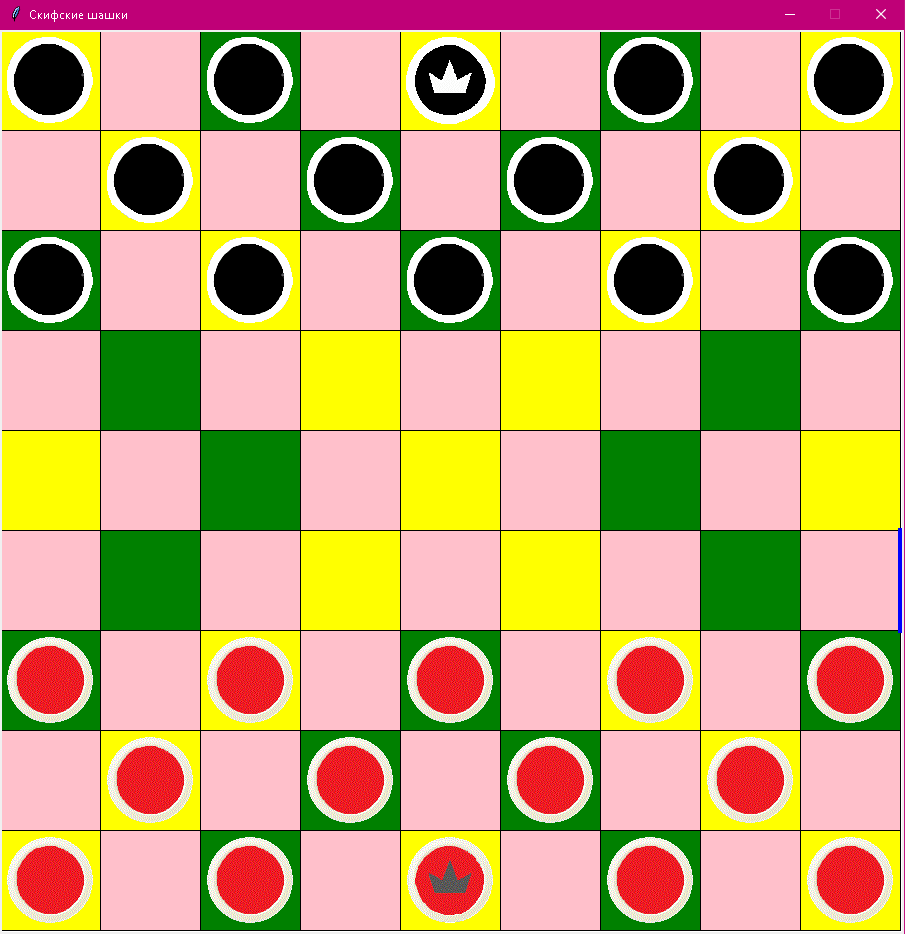
Порядок работы:

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

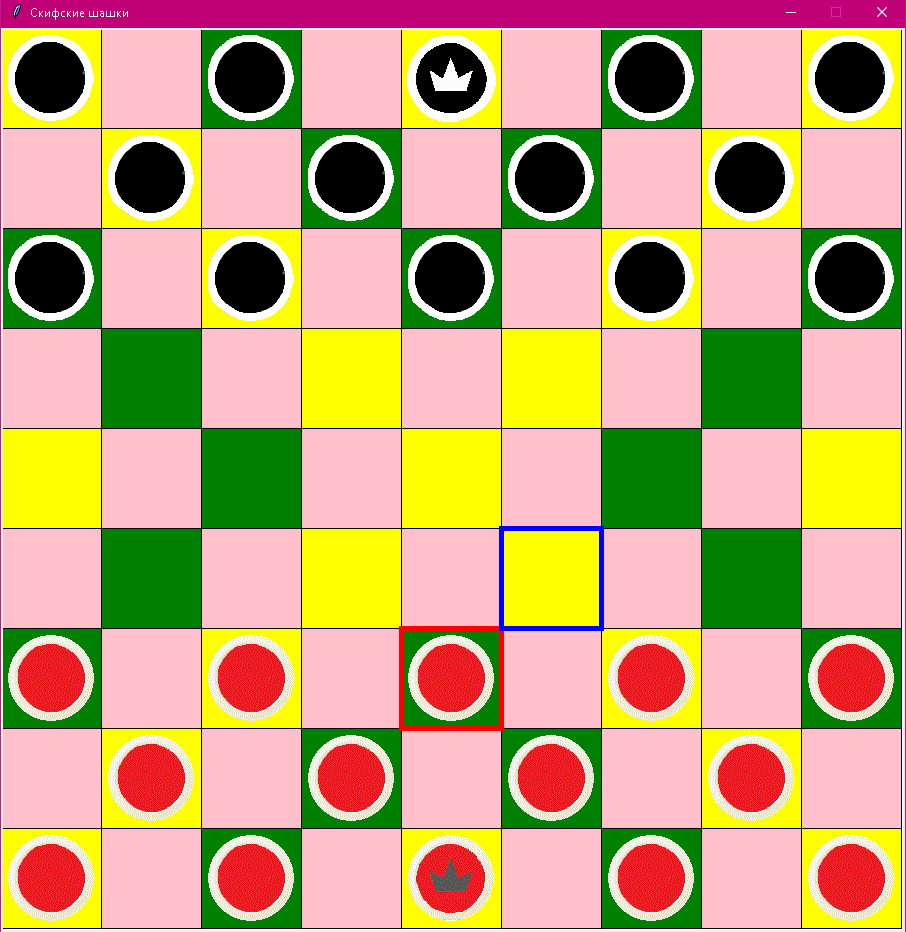


После введения данных и нажатия на кнопку «Зарегистрироваться» появляется окно с текстом об успешной регистрации аккаунта и просьбой заново войти в свой аккаунт.

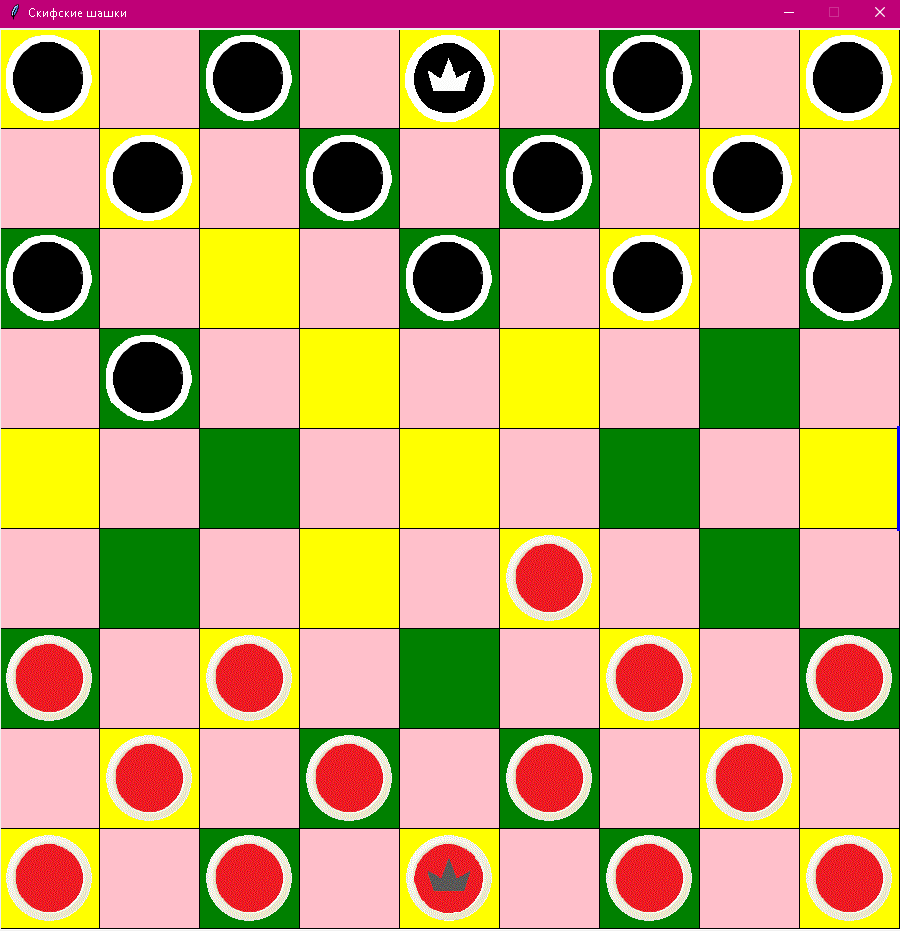
При успешной авторизации открывается окно игры.



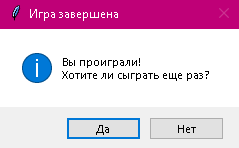
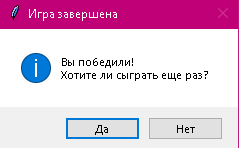
Далее пользователю следует левой кнопкой мыши выбрать шашку, которой он хочет пойти, и далее указать соседнюю клетку с ней для хода

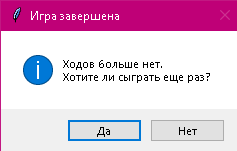


После хода белыми(красными) шашками, право хода приходит черным, ход будет делать компьютер.



После того как у кого-то из игроков закончились шашки, либо они заблокированы. Программа выдаст соответствующее сообщение (выиграли, проиграли, заблокированы)



Если нажмете ДА, то будет переигровка, если НЕТ, то выход.

**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\_parol» - алгоритм, отвечающий за хеширование пароля пользователя.
2. «enter\_users» - алгоритм, отвечающий за вход пользователя в игру
3. «vivod\_polya» – алгоритм, отвечающий за отрисовку игрового поля и шашек на ней.
4. «hod\_igroka» - алгоритм, обрабатывающий ход игрока.
5. «hod\_komp» - алгоритм, обрабатывающий ход компьютера.
6. «spisok\_hodov\_igroka» - алгоритм, составляющий список ходов игрока
7. «spisok\_hodov\_komp» - алгоритм, составляющий список ходов компьютера
8. «proverka\_hodov\_komp» - алгоритм, проверяющий наличие ходов компьютера и подбирающий наилучший ход
9. «podschet» - 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 ли сыграть еще раз?» или «Вы проиграли! Хотите ли сыграть еще раз?» или «Ходов больше нет. Хотите ли сыграть еще раз?».

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 «Логин и Пароль должны быть заполнены».

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

При успешной регистрации откроется окно с сообщением «Регистрация успешно завершена, войдите в аккаунт »

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

**Полный код программы:**

from tkinter import \*

from tkinter import messagebox

import random

import time

import copy

import os

import hashlib

def hash\_parol(parol): # Хеширование пароля

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

def registr\_user(): # Регистрируем пользователя

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

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

elif proverka\_logina():

messagebox.showerror("Ошибка", "Учетная запись уже существует.")

else:

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

file.write(f"{login.get()}:{hash\_parol(parol.get())}\n")

messagebox.showinfo("Успех","Регистрация успешно завершена.\nВойдите в аккаунт")

def proverka\_logina(): # Проверяем наличие логина в файле

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

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

lines = file.readlines()

login\_vvod = login.get()

for line in lines:

if login\_vvod in line:

return True

return False

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

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

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

lines = file.readlines()

login\_vvod = login.get()

parol\_vvod = parol.get()

for line in lines:

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

if len(parts) == 2:

sohranenii\_login, sohranenii\_parol = parts

if login\_vvod == sohranenii\_login and hash\_parol(parol\_vvod) == sohranenii\_parol:

return True

return False

def enter\_users(): # Атвторизуем пользователя

if proverka\_users():

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

root.destroy()

else:

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

root = Tk() # Создаем окно

root.title("Регистрация/вход")

root.geometry("400x300")

Label\_login = Label(text="Логин")

Label\_login.pack(padx=6, pady=6)

login = Entry(bd=2)

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

Label\_parol = Label(text="Пароль")

Label\_parol.pack(padx=6, pady=6)

parol = Entry(bd=2)

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

vhod\_btn1 = Button(text="Войти",command=enter\_users)

vhod\_btn1.pack(padx=6, pady=6)

registr\_btn2 = Button(text="Зарегистрироваться",command=registr\_user)

registr\_btn2.pack(padx=6, pady=6)

root.mainloop()

gl\_okno=Tk()#создаём окно

gl\_okno.resizable(width = False, height = False)

gl\_okno.title('Скифские шашки')#заголовок окна

doska=Canvas(gl\_okno, width=900,height=900,bg='#FFFFFF')

doska.pack()

spisok\_hodov\_komputera = ()#конечный список ходов компьютера

predskazanie\_hodi = 3#количество предсказываемых компьютером ходов

k\_rezult=0#!!!

o\_rezult=0

poz1\_x=-1#клетка не задана

opredelitel\_hoda\_igroka=True#определение хода игрока(да)

def izobrazheniya\_shashek():#загружаем изображения шашек

global shashki

i1=PhotoImage(file="res\\whiter.gif")

i2=PhotoImage(file="res\\whiteq.gif")

i3=PhotoImage(file="res\\blackr.gif")

i4=PhotoImage(file="res\\blackq.gif")

shashki=[0,i1,i2,i3,i4]

def novaya\_igra():#начинаем новую игру

global pole

pole=[[3, 0, 3, 0, 4, 0, 3, 0, 3],

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

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

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

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

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

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

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

[1, 0, 1, 0, 2, 0, 1, 0, 1]]

def vivod\_polya(x\_poz\_1, y\_poz\_1, x\_poz\_2, y\_poz\_2):#рисуем игровое поле

global shashki

global pole

global kr\_ramka,sin\_ramka

k=100

x=0

doska.delete('all')

while x<9\*k:#рисуем доску

y=1\*k

while y<9\*k:

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

y+=2\*k

x+=2\*k

x=1\*k

while x<9\*k:

y=0

while y<9\*k:

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

y+=2\*k

x+=2\*k

x=0

while x<9\*k:

y=0

while y<9\*k:

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

y+=4\*k

x+=4\*k

x=1 \* k

while x<9\*k:

y=1\*k

while y<9\*k:

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

y+=6\*k

x+=6\*k

x = 2 \* k

while x < 9 \* k:

y = 2 \* k

while y < 9 \* k:

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

y += 4 \* k

x += 4 \* k

x = 3 \* k

while x < 7 \* k:

y = 3 \* k

while y < 7 \* k:

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

y += 2 \* k

x += 2 \* k

x = 0

while x < 9 \* k:

y = 2 \* k

while y < 9 \* k:

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

y += 4 \* k

x += 8 \* k

x = 2\*k

while x < 9 \* k:

y = 0

while y < 9 \* k:

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

y += 8 \* k

x += 4 \* k

x = 1 \* k

while x < 9 \* k:

y = 3\*k

while y < 7 \* k:

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

y += 2 \* k

x += 6 \* k

x = 3 \* k

while x < 7 \* k:

y = 1 \* k

while y < 8 \* k:

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

y += 6 \* k

x += 2 \* k

x = 4 \* k

while x < 7 \* k:

y = 2 \* k

while y < 8 \* k:

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

y += 4 \* k

x += 6 \* k

x = 2 \* k

while x < 7 \* k:

y = 4 \* k

while y < 8 \* k:

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

y += 8 \* k

x += 4 \* k

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

sin\_ramka = doska.create\_rectangle(-5, -5, -5, -5, outline="blue", width=5)

for y in range(9):#рисуем шашки

for x in range(9):

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

if z:

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

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

z=pole[y\_poz\_1][x\_poz\_1] #рисуем активную шашку

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=shashki[z],tag='ani')

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

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

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

for ii in range(33):

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

doska.update()

time.sleep(0.01)

def soobsenie(s): #сообщение об окончании игры

global opredelitel\_hoda\_igroka

z='Игра завершена'

if s==1:

i=messagebox.askyesno(title=z, message='Вы победили!\nХотите ли сыграть еще раз?',icon='info')

if s==2:

i=messagebox.askyesno(title=z, message='Вы проиграли!\nХотите ли сыграть еще раз?',icon='info')

if s==3:

i=messagebox.askyesno(title=z, message='Ходов больше нет.\nХотите ли сыграть еще раз?',icon='info')

if i:

novaya\_igra()

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

opredelitel\_hoda\_igroka=True#ход игрока доступен

if not i:

gl\_okno.destroy()

def pozici\_1(event):#выбор клетки для хода 1

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

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

def pozici\_2(event):#выбор клетки для хода 2

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

global opredelitel\_hoda\_igroka

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

if pole[y][x]==1 or pole[y][x]==2:#проверяем шашку игрока в выбранной клетке

doska.coords(kr\_ramka,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 opredelitel\_hoda\_igroka:#ход игрока

hod\_igroka()

if not(opredelitel\_hoda\_igroka):

time.sleep(0.5)

hod\_komp()#передаём ход компьютеру

poz1\_x=-1#клетка не выбрана

def hod\_komp():

global opredelitel\_hoda\_igroka

global spisok\_hodov\_komputera

proverka\_hodov\_komp(1, (), [])

if spisok\_hodov\_komputera:#проверяем наличие доступных ходов

kol\_hodov=len(spisok\_hodov\_komputera)#количество ходов

th=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\_komputera`

deep\_hoda=len(spisok\_hodov\_komputera[th])#вычисляем длину выбранного случайного хода из списка `spisok\_hodov\_komputera`

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

peredvizenie\_shashki(1, spisok\_hodov\_komputera[th][i][0], spisok\_hodov\_komputera[th][i][1], #выполняем ход

spisok\_hodov\_komputera[th][1 + i][0], spisok\_hodov\_komputera[th][1 + i][1])

spisok\_hodov\_komputera=[]#очищаем список ходов

opredelitel\_hoda\_igroka=True#ход игрока доступен

s\_komp,s\_igrok=podschet() #определяем победителя

if not(s\_igrok):

soobsenie(2)

elif not(s\_komp):

soobsenie(1)

elif opredelitel\_hoda\_igroka and not(spisok\_hodov\_igroka()):

soobsenie(3)

elif not(opredelitel\_hoda\_igroka) and not(spisok\_hodov\_komp()):

soobsenie(3)

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

spisok=prosmotr\_hodov\_k1([])#здесь проверяем обязательные ходы

if not(spisok):

spisok=prosmotr\_hodov\_k2([])#здесь проверяем оставшиеся ходы

return spisok

def proverka\_hodov\_komp(tekh\_hod, n\_spisok, spisok):

global pole

global spisok\_hodov\_komputera

global l\_rezult,k\_rezult,o\_rezult

if not(spisok): #если список ходов пустой

spisok=spisok\_hodov\_komp() #заполняем

if spisok:

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

for ((poz1\_x,poz1\_y),(poz2\_x,poz2\_y)) in spisok:#проходим все ходы по списку

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

if t\_spisok:#если существует ещё ход

proverka\_hodov\_komp(tekh\_hod, (n\_spisok + ((poz1\_x, poz1\_y),)), t\_spisok)

else:

proverka\_hodov\_igroka(tekh\_hod, [])

if tekh\_hod==1:

t\_rez=o\_rezult/k\_rezult

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

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

l\_rezult=t\_rez#сохряняем наилучший результат

else:

if t\_rez==l\_rezult:

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

if t\_rez>l\_rezult:

spisok\_hodov\_komputera=()

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

l\_rezult=t\_rez#сохряняем наилучший результат

o\_rezult, k\_rezult = 0, 0

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

else:

s\_komp,s\_igrok=podschet()#подсчёт результата хода

o\_rezult+=(s\_komp-s\_igrok)

k\_rezult+=1

def spisok\_hodov\_igroka():#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 proverka\_hodov\_igroka(tekh\_hod, spisok):

global pole,k\_rezult,o\_rezult

global predskazanie\_hodi

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:

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

if t\_spisok:#если существует ещё ход

proverka\_hodov\_igroka(tekh\_hod, t\_spisok)

else:

if tekh\_hod<predskazanie\_hodi:

proverka\_hodov\_komp(tekh\_hod + 1, (), [])

else:

s\_komp,s\_igrok=podschet()#подсчёт результата хода

o\_rezult+=(s\_komp-s\_igrok)

k\_rezult+=1

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

else:#доступных ходов нет

s\_komp,s\_igrok=podschet()#подсчёт результата хода

o\_rezult+=(s\_komp-s\_igrok)

k\_rezult+=1

def podschet():#подсчёт шашек на поле

global pole

s\_igrok=0

s\_komp=0

for i in range(9):

for ii in pole[i]:

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

if ii==2:s\_igrok+=2

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

if ii==4:s\_komp+=2

return s\_komp,s\_igrok

def hod\_igroka():

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

global opredelitel\_hoda\_igroka

opredelitel\_hoda\_igroka=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 соответствие правилам игры

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

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

opredelitel\_hoda\_igroka=True#считаем ход игрока невыполненным

else:

opredelitel\_hoda\_igroka=True#считаем ход игрока невыполненным

doska.update()

def peredvizenie\_shashki(f, poz1\_x, poz1\_y, poz2\_x, poz2\_y):

global pole

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

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

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

if poz2\_y==8 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

kx=ky=1 #перепрыгиваем шашку игрока

if poz1\_x<poz2\_x:kx=-1 # если координаты x позиции начальной шашки меньше координаты x позиции конечной шашки

if poz1\_y<poz2\_y:ky=-1 # если координаты y позиции начальной шашки меньше координаты y позиции конечной шашки

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): # переменные x\_poz и y\_poz устанавливаются равными координатам конечной позиции шашки

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:vivod\_polya(-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:#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:#игрока

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

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

def prosmotr\_hodov\_k1(spisok):#проверка наличия обязательных ходов

for y in range(9):#сканируем всё поле

for x in range(9):

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

return spisok

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

if pole[y][x]==3:#шашка

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

if 0<=y+iy+iy<=8 and 0<=x+ix+ix<=8:

if pole[y+iy][x+ix]==1 or pole[y+iy][x+ix]==2: # 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\_hod=0#определение правильности хода

for i in range(1,9):

if 0<=y+iy\*i<=8 and 0<=x+ix\*i<=8:

if true\_hod==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:

true\_hod+=1

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

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

break

return spisok

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

for y in range(9):#сканируем всё поле

for x in range(9):

if pole[y][x]==3:#шашки

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

if 0<=y+iy<=8 and 0<=x+ix<=8:

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:

if 0<=y+iy\*2<=8 and 0<=x+ix\*2<=8:

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\_hod=0#определение правильности хода

for i in range(1,9):

if 0<=y+iy\*i<=8 and 0<=x+ix\*i<=8:

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:

true\_hod+=1

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

break

return spisok

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

spisok=[]#список ходов

for y in range(9):#сканируем всё поле

for x in range(9):

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

return spisok

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

if pole[y][x]==1:#шашки

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

if 0<=y+iy+iy<=8 and 0<=x+ix+ix<=8:

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

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\_hod=0#определение правильности хода

for i in range(1,9):

if 0<=y+iy\*i<=8 and 0<=x+ix\*i<=8:

if true\_hod==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:

true\_hod+=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 true\_hod==2: #если на пути встречается своя фигура

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

break

return spisok

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

for y in range(9): #сканируем всё поле

for x in range(9):

if pole[y][x]==1: #шашка

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

if 0<=y+iy<=8 and 0<=x+ix<=8:

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:

if 0<=y+iy\*2<=8 and 0<=x+ix\*2<=8:

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\_hod=0 #определение правильности хода

for i in range(1,9):

if 0<=y+iy\*i<=8 and 0<=x+ix\*i<=8:

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:

true\_hod+=1

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

break

return spisok

izobrazheniya\_shashek() # здесь загружаем изображения шашек

novaya\_igra() # начинаем новую игру

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

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

doska.bind("<Button-1>", pozici\_2) # нажатие левой кнопки

mainloop()