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

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

по дисциплине «Алгоритмы и структуры данных»

# Тема:Компьютерная логическая игра «Международные шашки - Поддавки»

Р.02069337.22/255-57 ПЗ-01

Листов 46

Руководитель разработки:

доцент кафедры ИВК *Шишкин Вадим Викторинович*

« » 2024 г.

Исполнитель:

студент гр. ИСТбд-21

*Рябов Айрат Андреевич*

« » 2024 г.

2024

Содержание

Аннотация……...…………………………………………………………3

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

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

Руководство программиста……………………………….…………...15

Текст программы…..…………………………………………………...21

**Аннотация**

Данный документ представляет собой пояснительную записку на курсовую работу на тему «Компьютерная логическая игра «Международные шашки – Поддавки». Документ содержит следующие разделы: техническое задание, пояснительная записка и руководство программиста, код программы; в нём излагается постановка задачи и описание реализуемой программы, её 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 “Международные шашки – поддавки”»

Р.02069337.22/255-57 ТЗ-01

Листов 4

**Исполнитель**:

студент гр. ИСТбд-21

*Рябов Айрат Андреевич*

« » 2024 г.

**2024**

**Введение**

Наименование разрабатываемого приложения – Компьютерная логическая игра «Международные шашки – Поддавки»

Приложение должно соответствовать следующим правилам игры. Игра ведется между двумя соперниками(пользователь-пользователь) шашками разного цвета на квадратном поле размером 10х10 клеток.Шашки расставляются на чёрных полях первых четырёх горизонтальных рядов с каждой стороны. Играющий синими ходит первый, далее ходы делаются поочередно. Шашки делятся на простые и дамки. В начальном положении все шашки простые.Правила хода:

1) Простая шашка ходит по диагонали вперёд на одну клетку. При достижении любого поля последней горизонтали, простая шашка превращается в дамку.

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

Взятие обязательно, если оно возможно.Если простая шашка в процессе взятия достигает «дамочного» поля и может бить дальше в роли простой шашки, то она этим ходом продолжает бой и остаётся простой. В противном случае она превращается в дамку и останавливается. Право боя по правилам дамки она приобретает лишь со следующего хода.

Партия считается выигранной ,если один из соперников отдаст все свои шашки, либо позволит сопернику запереть их. Победитель-тот, кто отдал все шашки или соперник запер их.

**1. Основания для разработки**

В качестве оснований для разработки учебный план направления 09.03.02 «Информационные системы и технологии» и распоряжение по факультету о закреплении тем курсовых работ (проектов) за студентами 2 курса ФИСТ направления 09.03.02 “Информационные системы и технологии” (профиль Информационные системы и технологии) по дисциплине “Алгоритмы и структуры данных”.

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

**2.1. Функциональное назначение**

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

**2.2 Требования к функциональным характеристикам**

Приложение должно предоставлять:

* Давать возможность поочередно ходить синими и красными шашками.
* Отображать информацию об этапе игры.
* Правильно графически отображать ходы на доске.
* Проверять правильность ходов в соответствии с правилами и не давать возможность делать не правильные ходы.
* Проверять окончание игры и определять победителя.

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

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

- регистрация/авторизация пользователя;

- отрисовка игрового поля;

- взаимодействие с пользователем;

- прием ,проверка правильности и отрисовка хода пользователя;

- проверка окончания игры;

- информирование пользователя об окончании игры и победителе.

2.2.2 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. Логин и пароль должны храниться в отдельном файле.

**2.3 Требования к надёжности**

Поддержка непрерывной и стабильной работы компьютера.

**2.4 Требования к информационной и программной совместимости**

Рекомендуется к использованию на Windows 7, 10.

При создании программы используются библиотекa “tkinter”.

Разработка ведётся в “Visual Studio Code” на версии языка программирования Python 3.9.

**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

Обеспечение свободного доступа к проекту в репозитории до окончания срока учебы.

2.6 3 Сроки хранения

Срок хранения – до окончания учебы**.**

**3. Требования к программной документации**

Определяются заданием на курсовую работу.

**4. Стадии и этапы разработки**

Определяются заданием на курсовую работу.

**5. Порядок контроля и приёмки**

Определяются заданием на курсовую работу.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

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

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

**Пояснительная записка**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/255-57 ПЗ-02

Листов 5

Исполнитель:

студент гр. ИСТбд-21

*Рябов Айрат Андреевич*

« » 2024 г.

2024

**Введение**

Наименование игры: Международые шашки – Поддавки

Условное наименование: international checkers

На 100-клеточной доске соперники расставляют свои 20 шашек на черных клетках на 4 первых горизонталях. Шашка может ходить только на 1 клетку влево-вверх, или вправо-вверх. Взятие шашки соперника обязательно и возможно, если за ней есть свободное поле. Если после взятия возможно продолжить брать шашки противника, то игрок обязан это сделать. Ходить назад простая шашка не может, зато может бить. Шашка, дошедшая до последней горизонтали, становится дамкой, но только если она дошла не после боя, если шашка бьет и доходит до горизонтали, но при этом может еще бить,она продолжает бой. Дамка в международных шашках ходит и ест в любом направлении по правилам. Цель игры в международные шашки - поддавки — лишить себя возможности ходить. Чаще всего это достигается поеданием всех шашек оппонентом.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется общей постановкой задачи в задании на курсовую работу.

**1.2 Математические методы**

Математические методы не применялись.

**1.3 Алгоритмы**

**1.3.1 Алгоритм проверка хода игрока**

Данный алгоритм необходим для того, чтобы пользователь не смог сделать некорректный ход. То есть ход можно сделать, основываясь на правила игры, иначе ход сделать не получится.

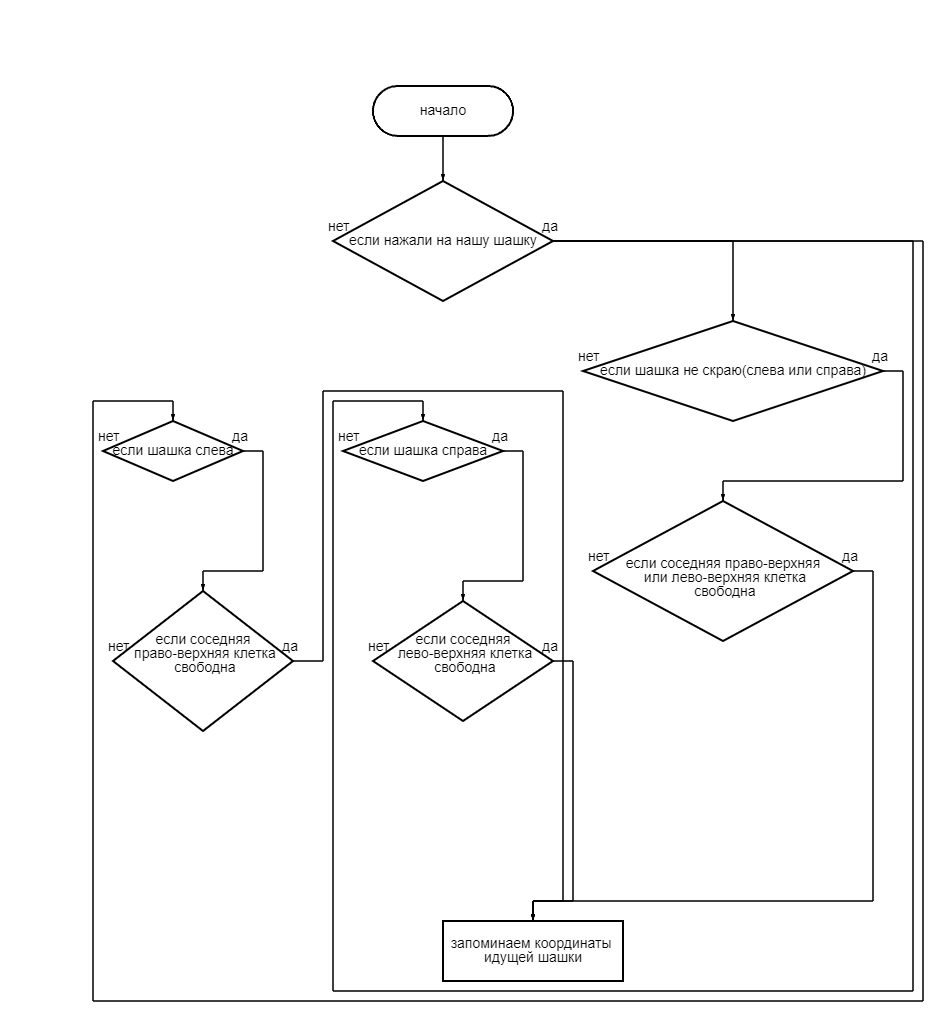


Рисунок 1 – Проверка хода игрока

**1.3.2 Алгоритм проверки боя.**

Данный алгоритм основной в игре. Идет проверка по всему полю , если находится случай ,что можно бить(по диагонали вначале бьющая шашка ,следом за ней противоположная и далее пустая клетка),в массив добавляются координаты бьющей шашки ,координаты бьющийся шашки и координаты пустой клетки для дальнейшей рубки.Этот массив добавляется еще в один массив,так как одну шашку могут срубить одновременно две шашки,и пользователь может выбрать какой шашкой ему бить.Если при проверке встречается дамка,то начинает работу следующий алгоритм.

**1.3.3 Алгоритм проверки боя для дамки.**

Для дамки нужно делать проверку во все 4 стороны. Для этого организуется 4 цикла по одному на каждую сторону. Если по пути следования цикла все поля пустые и далее встречается вражеская шашка ,а за ней пустая клетка , либо сразу вражеская шашка и пустая клетка, флаг взятия становиться истинным и начинают работу еще 2 цикла, идущие влево и вправо от идущей за вражеской шашкой клеткой. Если в любом из двух этих циклов встречается вариант дальнейшего взятия ,включается еще один флаг ,цикл останавливается. Далее если второй флаг истинный ,то в конечный массив для боя добавляются только координаты тех клеток ,откуда можно бить дальше. Иначе добавляются все пустые клетки после вражеской шашки(в любую из них может встать дамка).

**1.4 Тестирование**

Тестирование программы – это этап, на котором проверяется, как ведет себя программа на как можно большем количестве входных наборов данных, в том числе и на заведомо неверных.

Тестирование – это процесс выполнения программы с целью выявления ошибок. Процесс разработки ПО предполагает три стадии тестирования:

* автономное тестирование – это тестирование компонентов ПО;
* комплексное тестирование;
* системное (оценочное) тестирование – тестирование на соответствие основным критериям качества.

Принципы тестирования:

* избегать тестирования программы самим автором;
* предполагаемые результаты должны быть известны до тестирования;
* необходимо изучать результаты каждого теста;
* необходимо проверять действие программы на неверных данных.

Существует два принципиально различных подхода к формированию тестов:

* структурный – известна структура тестируемого ПО, в том числе его алгоритмы. Тесты строят так, чтобы проверить правильность реализации заданной логики в ходе программы (белый ящик);
* функциональный – структура ПО неизвестна. Тесты строят по функциональным спецификациям (черный ящик; подход, управляемый данными).

При проведении тестирования следует помнить, что никакое тестирование не может доказать отсутствие ошибок в ПО. Удачным считают тест, который обнаруживает хотя бы одну ошибку. Вероятность наличия необнаруженных ошибок пропорциональна количеству уже найденных ошибок в программе.

**1.4.1 Методика проведения и результаты тестирования**

При разработке данной программы были допущены следующие синтаксические ошибки:

* неправильное использование операторов присваивания;
* неверное объявление циклов.

При тестировании были выполнены следующие принципы:

* тщательность подбора данных для теста программы, не только для правильных входных данных, но и для неправильных;
* доскональное изучение результатов тестирования.

**1.4.2 Отладка выявленных ошибок, обнаруженных при тестировании**

Во время выполнения отладки, были использованы следующие методы:

* анализ кода без исполнения программы;
* запуск программы из под отладчика.

Все синтаксические ошибки были исправлены при компиляции проекта, учитывая синтаксические особенности среды программирования. Также в программном средстве возникали логические ошибки, которые были исправлены путем пересмотра кода алгоритмов и его последующего исправления.

Отладка ряда ошибок прошла успешно, были внесены изменения в интерфейс программы, а также были предусмотрены некоторые исключительные ситуации.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

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

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

**Руководство программиста**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/255-57 РП-03

Листов 5

Исполнитель:

студент гр. ИСТбд-21

*Рябов Айрат Андреевич*

« » 2024 г.

2024

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложение предназначено для развлечения, развития интеллекта и стратегического мышления.

Правила игры:

* Положение фигур полностью симметрично.
* У обоих игроков по 20 шашек.
* Любые шашки, попадая на последнюю противоположную горизонталь, становятся дамками.

Функциональные возможности

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

В приложении реализованы: главное меню, окно регистрации/авторизации, модуль, который отрисовывает игру и проверяет на выигрыш/ничью, меню для создания новой партии или выхода из приложения

**1.2 Условия, необходимые для использования приложения**

1. Операционная система: Приложение должно быть доступно для Windows.

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: Приложение использует библиотеки, такие как tkinter для создания графического интерфейса пользователя, sys ,для выхода из приложения,а также стандартные библиотеки языка Python версии 3. и выше

**2. Характеристики программы**

**2.1 Характеристики приложения**

Количество значимых строк: 1050

Количество структур данных: 3(структура для хранения состояния шахматной доски и соответствующими методами поиска допустимых ходов, проверки доски на выигрыш/ничью, добавления хода и другие, структура для хранения состояния шахматной доски, которая может прочитаться алгоритмом отрисовки интерфейса, структура данных для хранения логина и пароля)

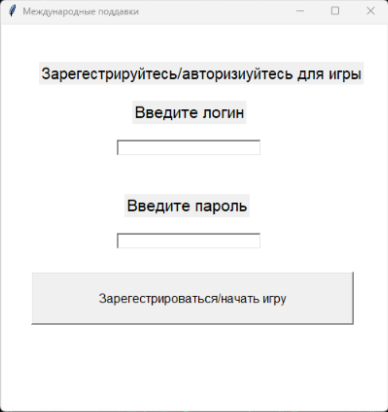
Количество алгоритмов: 4(алгоритм для отрисовки игрового поля и возможных ходов, алгоритм проверки возможности хода простой шашкой, алгоритм проверки возможности хода дамкой с дальнейшим сохранением возможных координат хода, алгоритм проверки на возможность взятия простой шашкой и дамкой)

Описание используемых библиотек.

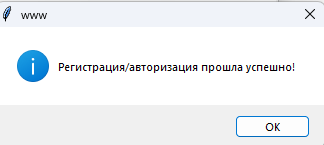
1. Библиотека tkinter: это стандартная библиотека для создания графического интерфейса пользователя (GUI) в Python. Она предоставляет различные виджеты и методы для создания окон, кнопок, полей ввода, меню и других элементов интерфейса. Tkinter обеспечивает простой способ взаимодействия с пользователем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

Описание работы приложения:

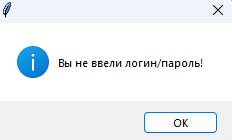
1. Изначально пользователь видит окно авторизации и регистрации, в котором можно зарегистрироваться или войти в существующий профиль.



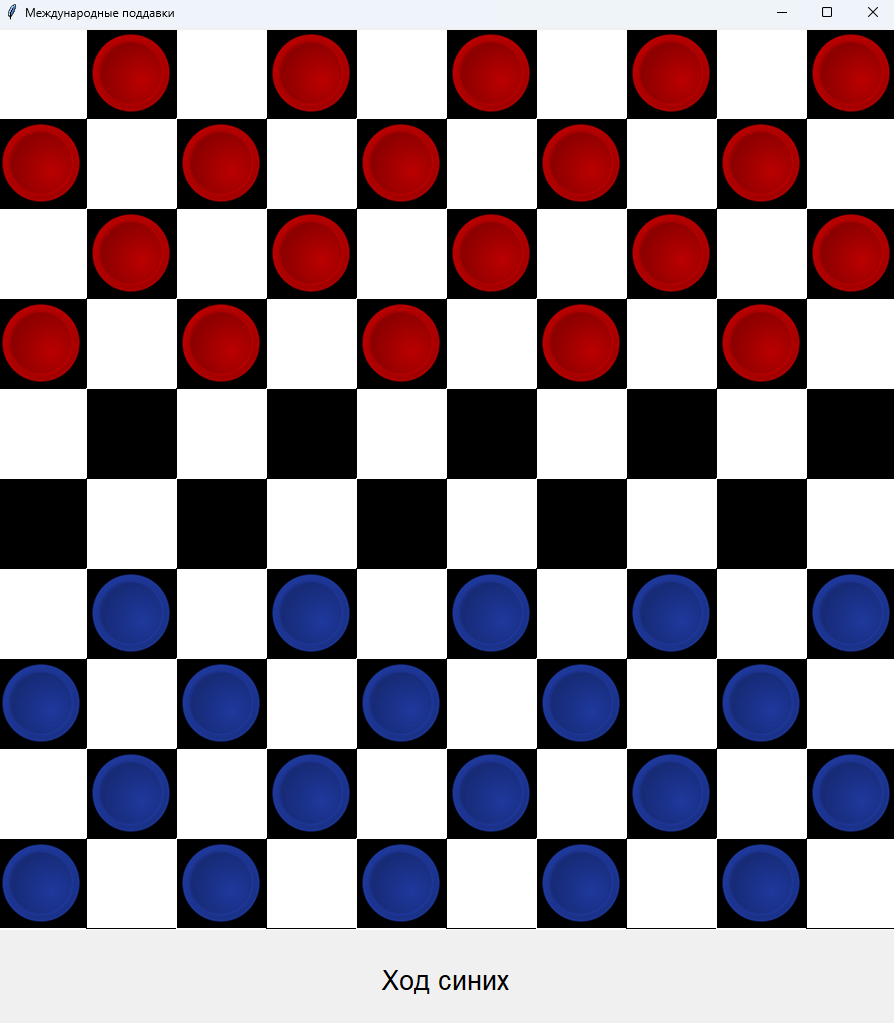
1. При нажатии на кнопку «Зарегистрироваться/начать игру» происходит регистрация/авторизация пользователя, появляется уведомление об успешной авторизации/регистрации.



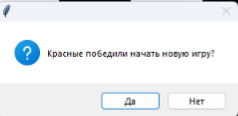
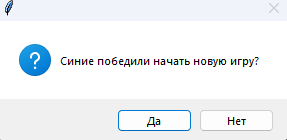
1. В окне регистрации/авторизации обрабатываются случаи правильного ввода данных(заполнены оба поля).В противном случае появляется сообщение об ошибки ввода данных.



1. Открывается игровое окно



1. После окончания игры (два случая) открывается соответствующее меню, в котором можно закрыть приложение или начать игру заново.

**2.2 Особенности реализации приложения**

Использован двумерный массив для хранения состояния доски, который может читаться алгоритмом создания графического игрового поля. Реализовано стандартными методами языка Python

Использована структура данных для хранения логина и пароля пользователя. Реализовано с помощью текстового документа. Реализованы методы проверки содержимого документа и добавление новых логинов и паролей.

**3. Обращение к программе**

New\_game() –сбрасывает первоначальные данные (флаги, поле, счетчики “Съеденных” шашек).

Register() – получает данные из полей ввода логина и пароля и проверяет /добавляет данные в базу данных(текстовый документ).

Win() – следит за количеством счетчиков съеденных шашек, информирует пользователей о победителе и проигравшем.

Board() – рисует игровое поле на основе двумерного массива,который является моделью игрового поля. В зависимости от значения на моделе ,добавляет на канвас нужное изображение шашки/дамки.

Vozm\_hod\_damka() – если пользователь желает сделать ход дамкой ,то этот метод проверяет возможность сделать ход ,в зависимости от игрока(красные / синие) и с помощью циклов(в право-вверх,в лево-вверх,в лево-низ, в право-низ) находит пустые клетки и добавляет их в двум.массив.

Vozm\_hoda\_bl() / Vozm\_hoda\_red() – проверяют возможность сделать обычный ход ,а так же сохраняют координаты шашки,которая хочет идти для дальнейшей правильной отрисовки хода.

Proverka\_bl() ,proverka\_red() – после хода/взятия одного из игроков вызываются эти методы ,для проверки на обязательное взятие.

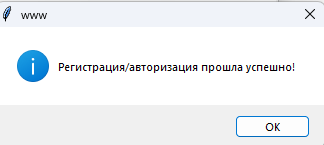
Dam\_bl(),dam\_red() – аналогично с тем,что выше ,только для дамок ,вызываются внутри предыдущих методов.

Click() – обработчик событий, который на основе модели игрового поля и различных структур (например массив с свободными для хода координатами клеток) меняет данные в модели игрового поля и далее вызывает метод board() для отрисовки нового поля.Таким образом после каждого хода/взятия меняется поле на экране у игрока.

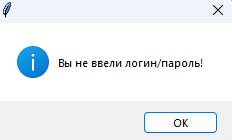
**4. Сообщения**

Сообщения выдаваемые по результатам контроля корректности ввода/вывода.

1.



2.



Полный код программы:

from tkinter import\*

from tkinter import font

from tkinter import messagebox

from tkinter.messagebox import askyesno

from PIL import ImageTk,Image

import sys

root = Tk()

root.geometry("500x500")

root.resizable(width=False, height=False)

root.title("Международные поддавки")

canvas = Canvas(root, width=900, height=900, bg='#fff')

canvas.pack()

im\_bl,im\_red,im\_bl\_d,im\_red\_d = ImageTk.PhotoImage(file="Blue.png"), ImageTk.PhotoImage(file="RED.png"),ImageTk.PhotoImage(file="Blue\_d.png"),ImageTk.PhotoImage(file="RED\_d.png")

shahiki = [0,im\_bl,im\_red,im\_bl\_d,im\_red\_d]

def new\_game():

global board

global c\_red,c\_bl,pole,hod\_bl,hod\_red,vozm\_hod\_bl,vozm\_hod\_red,vzatie\_bl ,vzatie\_red, hod\_dm\_bl,hod\_dm\_red,vzatie\_bl\_d

hod\_bl = True

hod\_red = False

vozm\_hod\_bl,vozm\_hod\_red = False,False

vzatie\_bl ,vzatie\_bl\_d,vzatie\_red = False,False,False

hod\_dm\_bl,hod\_dm\_red = False,False

pole = [[0,2,0,2,0,2,0,2,0,2],

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

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

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

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

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

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

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

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

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

c\_bl,c\_red = 20,20

board()

hod\_bl = True

hod\_red = False

vozm\_hod\_bl,vozm\_hod\_red = False,False

vzatie\_bl ,vz\_d,vzatie\_red = False,False,False

hod\_dm\_bl,hod\_dm\_red = False,False

c\_bl,c\_red = 20,20

lst3 = []

pole = [[0,2,0,2,0,2,0,2,0,2],

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

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

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

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

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

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

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

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

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

def register(): #Регистрация пользователя

global logg ,passs , txt,label

lg = logg.get()

ps = passs.get()

if len(lg) == 0 or len(ps) == 0:

return messagebox.showinfo(message="Вы не ввели логин/пароль!")

else:

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

a = file.read().split()

if len(a) > 0:

for i in range(len(a)):

if a[i] == lg and a[i+1] == ps:

break

else:

file.write(lg+' '+ps+' ')

else:

file.write(lg+' '+ps+' ')

file.close()

messagebox.showinfo('www',message="Регистрация/авторизация прошла успешно!")

txt.place\_forget()

txt0.place\_forget()

txt1.place\_forget()

bt.place\_forget()

logg.place\_forget()

passs.place\_forget()

root.geometry("900x1000")

root.geometry(f"+{(root.winfo\_screenwidth() - 700) // 2}+{(root.winfo\_screenheight() - 1050) // 2}")

root.resizable(width=False, height=False)

label = Label(root,text='Ход синих',width=900,height=100,font=("Arial", 20))

label.pack()

canvas.bind("<Button-1>",click)

board()

def win():

global c\_bl,c\_red

global new\_game

if c\_red == 0:

if askyesno(message='Красные победили начать новую игру?'):

new\_game()

else:

sys.exit()

if c\_bl == 0:

if askyesno(message='Синие победили начать новую игру?'):

new\_game()

else:

sys.exit()

def board(): #Отрисовка доски

global pole ,canvas

fill = '#fff'

outline = '#000'

k = 90

canvas.delete('all')

for i in range(10):

for j in range(10):

x1, y1, x2, y2 = i \* k, j \* k, i \* k + k, j \* k + k

canvas.create\_rectangle(x1, y1, x2, y2, fill=fill, outline=outline)

fill, outline = outline, fill

fill, outline = outline, fill

for i in range(10):

for j in range(10):

q = pole[j][i]

if q:

canvas.create\_image(i \* k, j \* k, anchor=NW, image=shahiki[q])

dm,dm1 = [],[]

def vozm\_hod\_damka(x,y):

global dm, hod\_dm\_bl,hod\_dm\_red,dm1

if pole[y][x] == 3:

dm = []

z = []

z.append(y)

z.append(x)

dm.append(z)

z = []

for i in range(1,11):

if y-i == -1 or x+i == 10:

break

if pole[y-i][x+i] == 0:

z.append(y-i)

z.append(x+i)

dm.append(z)

z = []

else:

break

for i in range(1,11):

if y-i == -1 or x-i == -1:

break

if pole[y-i][x-i] == 0:

z.append(y-i)

z.append(x-i)

dm.append(z)

z = []

else:

break

for i in range(1,11):

if y+i == 10 or x+i == 10:

break

if pole[y+i][x+i] == 0:

z.append(y+i)

z.append(x+i)

dm.append(z)

z = []

else:

break

for i in range(1,11):

if y+i == 10 or x-i == -1:

break

if pole[y+i][x-i] == 0:

z.append(y+i)

z.append(x-i)

dm.append(z)

z = []

else:

break

if len(dm) != 0:

hod\_dm\_bl = True

else:

hod\_dm\_bl = False

elif pole[y][x] == 4:

dm1 = []

z = []

z.append(y)

z.append(x)

dm1.append(z)

z = []

for i in range(1,11):

if y-i == -1 or x+i == 10:

break

if pole[y-i][x+i] == 0:

z.append(y-i)

z.append(x+i)

dm1.append(z)

z = []

else:

break

for i in range(1,11):

if y-i == -1 or x-i == -1:

break

if pole[y-i][x-i] == 0:

z.append(y-i)

z.append(x-i)

dm1.append(z)

z = []

else:

break

for i in range(1,11):

if y+i == 10 or x+i == 10:

break

if pole[y+i][x+i] == 0:

z.append(y+i)

z.append(x+i)

dm1.append(z)

z = []

else:

break

for i in range(1,11):

if y+i == 10 or x-i == -1:

break

if pole[y+i][x-i] == 0:

z.append(y+i)

z.append(x-i)

dm1.append(z)

z = []

else:

break

if len(dm1) != 0:

hod\_dm\_red = True

else:

hod\_dm\_red = False

else:

hod\_dm\_red = False

hod\_dm\_bl = False

def vozm\_hoda\_bl(x,y):

global vozm\_hod\_damka

global prev\_y\_bl,prev\_x\_bl,dm

global vozm\_hod\_bl

if pole[y][x] == 1:

dm = []

if 0 <= x <= 8:

if (pole[y-1][x+1] == 0 ) or (pole[y-1][x-1] == 0):

vozm\_hod\_bl = True

prev\_y\_bl = y

prev\_x\_bl = x

else:

vozm\_hod\_bl = False

if x == 9:

if pole[y-1][x-1] == 0:

vozm\_hod\_bl = True

prev\_y\_bl = y

prev\_x\_bl = x

else:

vozm\_hod\_bl = False

if x == 0:

if pole[y-1][x+1] == 0:

vozm\_hod\_bl = True

prev\_y\_bl = y

prev\_x\_bl = x

else:

vozm\_hod\_bl = False

def vozm\_hoda\_red(x,y):

global vozm\_hod\_red

global prev\_y\_red,prev\_x\_red

global vozm\_hod\_bl

if pole[y][x] == 2:

if 0 <= x <= 8:

if (pole[y+1][x+1] == 0) or (pole[y+1][x-1] == 0):

vozm\_hod\_red = True

prev\_y\_red = y

prev\_x\_red = x

else:

vozm\_hod\_red = False

if x == 0:

if pole[y+1][x+1] == 0:

vozm\_hod\_red = True

prev\_y\_red = y

prev\_x\_red = x

else:

vozm\_hod\_red = False

if x == 9:

if pole[y+1][x-1] == 0:

vozm\_hod\_red = True

prev\_y\_red = y

prev\_x\_red = x

else:

vozm\_hod\_red = False

l2 = []

f1 ,f2,f3,f4= False,False,False,False

def dam\_bl():

global vzatie\_bl\_d,vzatie\_bl,x\_p,y\_p,f1,l2,l3,f2

f1 ,f2,f3,f4= False,False,False,False

for i in range(10):

for j in range(10):

if pole[i][j] == 3:

for ii in range(1,11):

if i-ii == 0 or j+ii == 9 or i == 0 or j == 9:

break

if pole[i-ii][j+ii] == 0:

continue

if (pole[i-ii][j+ii] == 2 or pole[i-ii][j+ii] == 4) and pole[i-ii-1][j+ii+1] == 0:

l2.append([i,j])

l2.append([i-ii,j+ii])

l2.append([i-ii-1,j+ii+1])

vzatie\_bl = True

x\_p = i-ii-1

y\_p = j+ii+1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p -k

y = y\_p + k

if x == -1 or y == 10:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y-l == 0 or x == 0 or y == 0:

break

if pole[x-l][y-l] == 0:

continue

if (pole[x-l][y-l] == 2 or pole[x-l][y-l] == 4) and pole[x-l-1][y-l-1] == 0 and pole[x-l+1][y-l+1] == 0:

l3.append([x,y])

f1 = True

break

for l in range(1,11):

if x+l == 9 or y+l == 9 or x == 9 or y == 9:

break

if pole[x+l][y+l] == 0:

continue

if (pole[x+l][y+l] == 2 or pole[x+l][y+l] == 4) and pole[x+l+1][y+l+1] == 0 and pole[x+l-1][x+l-1] == 0:

l3.append([x,y])

f1 = True

break

if f1:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

#----------------------------------------------------------------------------------------------------------

for i in range(10):

for j in range(10):

if pole[i][j] == 3:

for ii in range(1,11):

if i-ii == 0 or j-ii == 0 or i == 0 or j == 0:

break

if pole[i-ii][j-ii] == 0:

continue

if (pole[i-ii][j-ii] == 2 or pole[i-ii][j-ii] == 4) and pole[i-ii-1][j-ii-1] == 0:

l2.append([i,j])

l2.append([i-ii,j-ii])

l2.append([i-ii-1,j-ii-1])

vzatie\_bl = True

x\_p = i-ii-1

y\_p = j-ii-1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p -k

y = y\_p - k

if x == -1 or y == -1:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y+l == 9 or x == 0 or y == 9:

break

if pole[x-l][y+l] == 0:

continue

if (pole[x-l][y+l] == 2 or pole[x-l][y+l] == 4) and pole[x-l-1][y+l+1] == 0 and pole[x-l+1][y+l-1] == 0:

l3.append([x,y])

f2 = True

break

for l in range(1,11):

if x+l == 9 or y-l == 0 or x == 9 or y == 0:

break

if pole[x+l][y-l] == 0:

continue

if (pole[x+l][y-l] == 2 or pole[x+l][y-l] == 4) and pole[x+l+1][y-l-1] == 0 and pole[x+l-1][y-l+1] == 0:

l3.append([x,y])

f2 = True

break

if f2:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

#---------------------------------------------------------------------------------------------------!-------

for i in range(10):

for j in range(10):

if pole[i][j] == 3:

for ii in range(1,11):

if i+ii == 9 or j-ii == 0 or i == 9 or j == 0:

break

if pole[i+ii][j-ii] == 0:

continue

if (pole[i+ii][j-ii] == 2 or pole[i+ii][j-ii] == 4) and pole[i+ii+1][j-ii-1] == 0:

l2.append([i,j])

l2.append([i+ii,j-ii])

l2.append([i+ii+1,j-ii-1])

vzatie\_bl = True

x\_p = i+ii+1

y\_p = j-ii-1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p +k

y = y\_p - k

if x == 10 or y == -1:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y-l == 0 or x == 0 or y == 0:

break

if pole[x-l][y-l] == 0:

continue

if (pole[x-l][y-l] == 2 or pole[x-l][y-l] == 4) and pole[x-l-1][y-l-1] == 0 and pole[x-l+1][y-l+1] == 0:

l3.append([x,y])

f3 = True

break

for l in range(1,11):

if x+l == 9 or y+l == 9 or x == 9 or y == 9:

break

if pole[x+l][y+l] == 0:

continue

if (pole[x+l][y+l] == 2 or pole[x+l][y+l] == 4) and pole[x+l+1][y+l+1] == 0 and pole[x+l-1][x+l-1] == 0:

l3.append([x,y])

f3 = True

break

if f3:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

#----------------------------------------------------------------------------------------------------------

for i in range(10):

for j in range(10):

if pole[i][j] == 3:

for ii in range(1,11):

if i+ii == 9 or j+ii == 9 or i == 9 or j == 9:

break

if pole[i+ii][j+ii] == 0:

continue

if (pole[i+ii][j+ii] == 2 or pole[i+ii][j+ii] == 4) and pole[i+ii+1][j+ii+1] == 0:

l2.append([i,j])

l2.append([i+ii,j+ii])

l2.append([i+ii+1,j+ii+1])

vzatie\_bl = True

x\_p = i+ii+1

y\_p = j+ii+1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p + k

y = y\_p + k

if x == 10 or y == 10:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y+l == 9 or x == 0 or y == 9:

break

if pole[x-l][y+l] == 0:

continue

if (pole[x-l][y+l] == 2 or pole[x-l][y+l] == 4) and pole[x-l-1][y+l+1] == 0 and pole[x-l+1][y+l-1] == 0:

l3.append([x,y])

f4 = True

break

for l in range(1,11):

if x+l == 9 or y-l == 0 or x == 9 or y == 0:

break

if pole[x+l][y-l] == 0:

continue

if (pole[x+l][y-l] == 2 or pole[x+l][y-l] == 4) and pole[x+l+1][y-l-1] == 0 and pole[x+l-1][y-l+1] == 0:

l3.append([x,y])

f4 = True

break

if f4:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

print(f1)

def proverka\_bl():

global dam\_bl

global vzatie\_bl ,lst3,vzatie\_bl\_d

global pole

lst2,lst1 = [],[]

for i in range(10):

for j in range(10):

if pole[i][j] == 1 and (i >= 2 and j <= 7):

if (pole[i-1][j+1] == 2 or pole[i-1][j+1] == 4) and pole[i-2][j+2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i-1)

lst2.append(j+1)

lst1.append(lst2)

lst2 = []

lst2.append(i-2)

lst2.append(j+2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_bl = True

if pole[i][j] == 1 and (j <= 7 and i <= 7):

if (pole[i+1][j+1] == 2 or pole[i+1][j+1] == 4) and pole[i+2][j+2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i+1)

lst2.append(j+1)

lst1.append(lst2)

lst2 = []

lst2.append(i+2)

lst2.append(j+2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_bl = True

if pole[i][j] == 1 and (j >= 2 and i >= 2):

if (pole[i-1][j-1] == 2 or pole[i-1][j-1] == 4) and pole[i-2][j-2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i-1)

lst2.append(j-1)

lst1.append(lst2)

lst2 = []

lst2.append(i-2)

lst2.append(j-2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_bl = True

if pole[i][j] == 1 and (j >= 2 and i <= 7):

if (pole[i+1][j-1] == 2 or pole[i+1][j-1] == 4) and pole[i+2][j-2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i+1)

lst2.append(j-1)

lst1.append(lst2)

lst2 = []

lst2.append(i+2)

lst2.append(j-2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_bl = True

if pole[i][j] == 3:

dam\_bl()

if len(lst3) == 0:

vzatie\_bl = False

def dam\_red():

global vzatie\_red,x\_p,y\_p,f1,l2,l3,f2

f1 ,f2,f3,f4= False,False,False,False

for i in range(10):

for j in range(10):

if pole[i][j] == 4:

for ii in range(1,11):

if i-ii == 0 or j+ii == 9 or i == 0 or j == 9:

break

if pole[i-ii][j+ii] == 0:

continue

if (pole[i-ii][j+ii] == 1 or pole[i-ii][j+ii] == 3) and pole[i-ii-1][j+ii+1] == 0:

l2.append([i,j])

l2.append([i-ii,j+ii])

l2.append([i-ii-1,j+ii+1])

vzatie\_red = True

x\_p = i-ii-1

y\_p = j+ii+1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p -k

y = y\_p + k

if x == -1 or y == 10:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y-l == 0 or x == 0 or y == 0 :

break

if pole[x-l][y-l] == 0:

continue

if (pole[x-l][y-l] == 1 or pole[x-l][y-l] == 3) and pole[x-l-1][y-l-1] == 0 and pole[x-l+1][y-l+1] == 0:

l3.append([x,y])

f1 = True

break

for l in range(1,11):

if x+l == 9 or y+l == 9 or x == 9 or y == 9:

break

if pole[x+l][y+l] == 0:

continue

if (pole[x+l][y+l] == 1 or pole[x+l][y+l] == 3) and pole[x+l+1][y+l+1] == 0 and pole[x+l-1][x+l-1] == 0:

l3.append([x,y])

f1 = True

break

if f1:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

#----------------------------------------------------------------------------------------------------!!!!------

for i in range(10):

for j in range(10):

if pole[i][j] == 4:

for ii in range(1,11):

if i-ii == 0 or j-ii == 0 or i == 0 or j == 0:

break

if pole[i-ii][j-ii] == 0:

continue

if (pole[i-ii][j-ii] == 1 or pole[i-ii][j-ii] == 3) and pole[i-ii-1][j-ii-1] == 0:

l2.append([i,j])

l2.append([i-ii,j-ii])

l2.append([i-ii-1,j-ii-1])

vzatie\_red = True

x\_p = i-ii-1

y\_p = j-ii-1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p -k

y = y\_p - k

if x == -1 or y == -1:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y+l == 9 or x == 0 or y == 9:

break

if pole[x-l][y+l] == 0:

continue

if (pole[x-l][y+l] == 1 or pole[x-l][y+l] == 3) and pole[x-l-1][y+l+1] == 0 and pole[x-l+1][y+l-1] == 0:

l3.append([x,y])

f2 = True

break

for l in range(1,11):

if x+l == 9 or y-l == 0 or x == 9 or y == 0:

break

if pole[x+l][y-l] == 0:

continue

if (pole[x+l][y-l] == 1 or pole[x+l][y-l] == 3) and pole[x+l+1][y-l-1] == 0 and pole[x+l-1][y-l+1] == 0:

l3.append([x,y])

f2 = True

break

if f2:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

#---------------------------------------------------------------------------------------------------!-------

for i in range(10):

for j in range(10):

if pole[i][j] == 4:

for ii in range(1,11):

if i+ii == 9 or j-ii == 0 or i == 9 or j == 0:

break

if pole[i+ii][j-ii] == 0:

continue

if (pole[i+ii][j-ii] == 1 or pole[i+ii][j-ii] == 3) and pole[i+ii+1][j-ii-1] == 0:

l2.append([i,j])

l2.append([i+ii,j-ii])

l2.append([i+ii+1,j-ii-1])

vzatie\_red = True

x\_p = i+ii+1

y\_p = j-ii-1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p +k

y = y\_p - k

if x == 10 or y == -1:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y-l == 0 or y == 0 or x == 0:

break

if pole[x-l][y-l] == 0:

continue

if (pole[x-l][y-l] == 1 or pole[x-l][y-l] == 3) and pole[x-l-1][y-l-1] == 0 and pole[x-l+1][y-l+1] == 0:

l3.append([x,y])

f3 = True

break

for l in range(1,11):

if x+l == 9 or y+l == 9 or x == 9 or y == 9:

break

if pole[x+l][y+l] == 0:

continue

if (pole[x+l][y+l] == 1 or pole[x+l][y+l] == 3) and pole[x+l+1][y+l+1] == 0 and pole[x+l-1][x+l-1] == 0:

l3.append([x,y])

f3 = True

break

if f3:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

#----------------------------------------------------------------------------------------------------------

for i in range(10):

for j in range(10):

if pole[i][j] == 4:

for ii in range(1,11):

if i+ii == 9 or j+ii == 9 or i == 9 or j == 9:

break

if pole[i+ii][j+ii] == 0:

continue

if (pole[i+ii][j+ii] == 1 or pole[i+ii][j+ii] == 3) and pole[i+ii+1][j+ii+1] == 0:

l2.append([i,j])

l2.append([i+ii,j+ii])

l2.append([i+ii+1,j+ii+1])

vzatie\_red = True

x\_p = i+ii+1

y\_p = j+ii+1

else:

break

l3 = l2.copy()

if len(l2) != 0:

for k in range(12):

x = x\_p + k

y = y\_p + k

if x == 10 or y == 10:

break

if pole[x][y] == 0:

l2.append([x,y])

for l in range(1,11):

if x-l == 0 or y+l == 9 or x == 0 or y == 9:

break

if pole[x-l][y+l] == 0:

continue

if (pole[x-l][y+l] == 1 or pole[x-l][y+l] == 3) and pole[x-l-1][y+l+1] == 0 and pole[x-l+1][y+l-1] == 0:

l3.append([x,y])

f4 = True

break

for l in range(1,11):

if x+l == 9 or y-l == 0 or y == 0 or x == 9:

break

if pole[x+l][y-l] == 0:

continue

if (pole[x+l][y-l] == 1 or pole[x+l][y-l] == 3) and pole[x+l+1][y-l-1] == 0 and pole[x+l-1][y-l+1] == 0:

l3.append([x,y])

f4 = True

break

if f4:

if len(l3) != 0:

lst3.append(l3)

else:

if len(l2) != 0:

lst3.append(l2)

l2,l3 = [],[]

def proverka\_red():

global dam\_red

global vzatie\_red ,lst3

global pole

lst2,lst1 = [],[]

for i in range(10):

for j in range(10):

if pole[i][j] == 2 and (i <= 7 and j >= 2):

if (pole[i+1][j-1] == 1 or pole[i+1][j-1] == 3 ) and pole[i+2][j-2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i+1)

lst2.append(j-1)

lst1.append(lst2)

lst2 = []

lst2.append(i+2)

lst2.append(j-2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_red = True

if pole[i][j] == 2 and (j >= 2 and i >= 2):

if (pole[i-1][j-1] == 1 or pole[i-1][j-1] == 3) and pole[i-2][j-2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i-1)

lst2.append(j-1)

lst1.append(lst2)

lst2 = []

lst2.append(i-2)

lst2.append(j-2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_red = True

if pole[i][j] == 2 and (j <= 7 and i >= 2):

if (pole[i-1][j+1] == 1 or pole[i-1][j+1] == 3) and pole[i-2][j+2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i-1)

lst2.append(j+1)

lst1.append(lst2)

lst2 = []

lst2.append(i-2)

lst2.append(j+2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_red = True

if pole[i][j] == 2 and (j <= 7 and i <= 7):

if (pole[i+1][j+1] == 1 or pole[i+1][j+1] == 3) and pole[i+2][j+2] == 0:

lst2.append(i)

lst2.append(j)

lst1.append(lst2)

lst2 = []

lst2.append(i+1)

lst2.append(j+1)

lst1.append(lst2)

lst2 = []

lst2.append(i+2)

lst2.append(j+2)

lst1.append(lst2)

lst2 = []

lst3.append(lst1)

lst1 = []

vzatie\_red = True

if pole[i][j] == 4:

dam\_red()

if len(lst3) == 0:

vzatie\_red = False

#----------------------------------------------------------------------------------------------------------------------------------

def click(event):

global hod\_bl,hod\_red,lst3,c\_bl,c\_red,dm,hod\_dm\_bl,dm1,l2,l3

global win, vozm\_hod\_damka

x,y = event.x//90 ,event.y//90

if hod\_bl:

if not vzatie\_bl:

label.config(text = 'Ход синих')

if pole[y][x] != 0:

vozm\_hoda\_bl(x,y)

vozm\_hod\_damka(x,y)

else:

if vozm\_hod\_bl and not hod\_dm\_bl :

if y != 0:

if (pole[y][x] == 0 and y +1 == prev\_y\_bl and x-1 == prev\_x\_bl) or (pole[y][x] == 0 and y +1 == prev\_y\_bl and x+1 == prev\_x\_bl):

pole[y][x] = 1

pole[prev\_y\_bl][prev\_x\_bl] = 0

proverka\_red()

hod\_bl = False

hod\_red = True

if y == 0:

if (pole[y][x] == 0 and y +1 == prev\_y\_bl and x-1 == prev\_x\_bl) or (pole[y][x] == 0 and y +1 == prev\_y\_bl and x+1 == prev\_x\_bl):

pole[y][x] = 3

pole[prev\_y\_bl][prev\_x\_bl] = 0

proverka\_red()

hod\_bl = False

hod\_red = True

if hod\_dm\_bl:

if pole[y][x] == 0 and ([y,x] in dm):

pole[y][x] , pole[dm[0][0]][dm[0][1]] = 3,0

dm = []

proverka\_red()

hod\_red = True

hod\_bl = False

if vzatie\_bl:

label.config(text='Обязательное взятие синих')

if pole[y][x] == 0:

for i in lst3:

if [y,x] in i:

if pole[i[0][0]][i[0][1]] == 1:

pole[i[0][0]][i[0][1]],pole[i[1][0]][i[1][1]],pole[y][x] = 0,0,1

if pole[i[0][0]][i[0][1]] == 3:

pole[i[0][0]][i[0][1]],pole[i[1][0]][i[1][1]],pole[y][x] = 0,0,3

c\_red -= 1

win()

lst3 = []

l3 ,l2 = [],[]

proverka\_bl()

if not vzatie\_bl:

if y == 0:

pole[y][x] = 3

hod\_bl = False

hod\_red = True

proverka\_red()

if hod\_red:

if not vzatie\_red:

label.config(text = 'Ход красных')

if pole[y][x] != 0:

vozm\_hoda\_red(x,y)

vozm\_hod\_damka(x,y)

else:

if vozm\_hod\_red and not hod\_dm\_red:

if y != 9:

if (pole[y][x] == 0 and y - 1 == prev\_y\_red and x-1 == prev\_x\_red) or (pole[y][x] == 0 and y - 1 == prev\_y\_red and x+1 == prev\_x\_red):

pole[y][x] = 2

pole[prev\_y\_red][prev\_x\_red] = 0

proverka\_bl()

hod\_red = False

hod\_bl = True

if vzatie\_bl:

label.config(text = 'Обязательное взятие синих')

if not vzatie\_bl:

label.config(text = 'Ход синих')

else:

if (pole[y][x] == 0 and y - 1 == prev\_y\_red and x-1 == prev\_x\_red) or (pole[y][x] == 0 and y - 1 == prev\_y\_red and x+1 == prev\_x\_red):

pole[y][x] = 4

pole[prev\_y\_red][prev\_x\_red] = 0

proverka\_bl()

hod\_red = False

hod\_bl = True

if vzatie\_bl:

label.config(text = 'Обязательное взятие синих')

if not vzatie\_bl:

label.config(text = 'Ход синих')

if hod\_dm\_red:

if pole[y][x] == 0 and ([y,x] in dm1):

pole[y][x] , pole[dm1[0][0]][dm1[0][1]] = 4,0

dm1 = []

proverka\_bl()

hod\_red = False

hod\_bl = True

if vzatie\_bl:

label.config(text='Обязательное взятие синих')

else:

label.config(text = 'Ход синих')

if vzatie\_red:

label.config(text='Обязательное взятие красных')

if pole[y][x] == 0:

for i in lst3:

if [y,x] in i:

if pole[i[0][0]][i[0][1]] == 2:

pole[i[0][0]][i[0][1]],pole[i[1][0]][i[1][1]],pole[i[2][0]][i[2][1]] = 0,0,2

if pole[i[0][0]][i[0][1]] == 4:

pole[i[0][0]][i[0][1]],pole[i[1][0]][i[1][1]],pole[y][x] = 0,0,4

c\_bl -= 1

win()

lst3 = []

proverka\_red()

if not vzatie\_red:

if y == 9:

pole[y][x] = 4

hod\_bl = True

hod\_red = False

proverka\_bl()

if not vzatie\_bl:

label.config(text = 'Ход синих')

else:

label.config(text = 'Обязательное взятие синих')

board()

txt0 = Label(root,text="Зарегестрируйтесь/авторизиуйтесь для игры",font=("Arial",15))

txt0.place(x=50,y=50)

txt = Label(root,text="Введите логин",font=("Arial",16))

txt.place(x=170 ,y=100)

logg = Entry(root,width=30,bd=2)

logg.place(x=150 ,y=150)

txt1= Label(root,text="Введите пароль",font=("Arial",16))

txt1.place(x=160 ,y=220)

passs = Entry(root,width=30,bd=2)

passs.place(x=150 ,y=270)

fnt=font.Font(size=12)

bt = Button(root,text="Зарегестрироваться/начать игру",width=45,height=3,font=fnt,command=register)

bt.place(x=40 ,y=320)

mainloop()