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 «Апит Содок»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/3457-49 ТЗ-03

Листов

**Руководитель разработки**:

доцент каф. ИВК, к.т.н., доцент

*Шишкин Вадим Викторинович*

« » 2023 г.

**Исполнитель**:

студент гр. ИСТбд-21

*Орешин Дмитрий Юрьевич*

« » 2023 г.

**2023**

Содержание

[**Аннотация** 3](#_Toc154708290)

[**Техническое задание** 4](#_Toc154708291)

[**Пояснительная записка** 9](#_Toc154708292)

[**Руководство программиста** 15](#_Toc154708293)

[**Текст программы:** 29](#_Toc154708294)

# **Аннотация**

Данный документ представляет собой пояснительную записку на курсовую работу на тему «Компьютерная логическая игра «Апит Содок»». 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 «Апит Содок»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/3457-49 ТЗ-03

Листов 5

**Исполнитель**:

студент гр. ИСТбд-21

*Орешин Дмитрий Юрьевич*

« » 2023 г.

**2023**

**Введение**

Разрабатываемое приложение наименования «Апит-Содок», реализует игру Апит Содок. Апит Содок - настольная логическая игра для двух игроков. Игроки поочередно ходят своими шашками на игровом поле. Во время хода игрок может передвинуть только одну свою шашку по горизонтали или вертикали на любое количество клеток, если на ее пути нет фигур. Цель – срубить все шашки противника.

**1. Основания для разработки**

Основание для разработки является учебный план направления 09.03.02 "Информационные системы и технологии", а также распоряжение по факультету.

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

**2.1. Функциональное назначение**

Требуется разработать однопользовательское десктопное приложение по игре в Апит Содок с графическим интерфейсом в среде Windows.

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

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

Приложение должно соответствовать следующим правилам игры.

**Поле и игроки.** Игра ведется между двумя соперниками (пользователь-компьютер) шашками разного цвета на квадратном поле размером 8х8 клеток. В начальной позиции у каждого игрока 16 шашек, построенных в первом и третьем ряду от игрока напротив шашек противника.

**Порядок ходов.** Игроки ходят по очереди. Первый ход делает пользователь или компьютер (в начале игры даётся выбор). Во время хода игрок может передвинуть только одну свою шашку по горизонтали или вертикали, как ладью в шахматах (то есть на любое количество клеток, если на ее пути нет фигур). Ход возможен во всех направлениях.

**Взятие шашек.** Возможны два типа взятия шашек противника: срубить шашку, взяв ее в клещи (окружить шашку противника двумя своими сверху и снизу или справа и слева) своими двумя шашками, или срубить сразу две шашки противника, встав между ними одной своей. Срубленные шашки сразу удаляются с доски.

**Цель игры.** Цель игры - срубить все шашки противника. Игра заканчивается, когда у одного из игроков не остается шашек. Игрок, у которого остались шашки, считается победителем, а второй у которого их нет - проигравшим.

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 ;

- интерактивные прием, проверка правильности и отрисовка хода пользователя;

- проверка окончания игры;

- вычисление, проверка правильности и отрисовка хода компьютера;

- информирование пользователя об окончании игры и победителе.

2.2.3 Требования к организации информационного обеспечения, входных и выходных данных

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

**2.3 Требования к надёжности**

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

**2.4 Требования к информационной и программной совместимости**

Операционная система: Windows 10.

Версия языка программирования: Python 3.10

Среда разработки: PyCharm Community Edition 2022.3.

При создании программы используются встроенные библиотеки “os”, “string”, “random” и сторонняя библиотека “tkinter 8.6.”.

**2.5 Требования к маркировке и упаковке**

Определяются заданием на курсовую работу.

**2.6 Требования к транспортированию и хранению**

2.6.1 Условия транспортирования

Требования к условиям транспортирования не предъявляются.

2.6 2 Условия хранения

Проект будет храниться в репозитории на сайте github.com по ссылке https://github.com/SiriRise/ApitSodok.git

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»**

# **Пояснительная записка**

**Тема:** Компьютерная логическая игра «Апит Содок»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/3457-49 ПЗ-03

Листов 6

**Исполнитель**:

студент гр. ИСТбд-21

*Орешин Дмитрий Юрьевич*

« » 2023 г.

**2023**

**Введение**

Приложение «Апит-Содок» реализует функционал логической игры «Апит Содок»

В качестве подхода для разработки была выбрана каскадная модель разработки («Водопад»). Каскадная модель была выбрана из-за своей простоты, она позволяет наглядно представить объём работы и сроки выполнения, а также эффективно разбить проект на несколько подзадач.

Приложение «Апит-Содок» представляет собой игру Апит Содок на 64 клеточном поле (8 клеток в ширину и 8 в длину) для двоих игроков.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

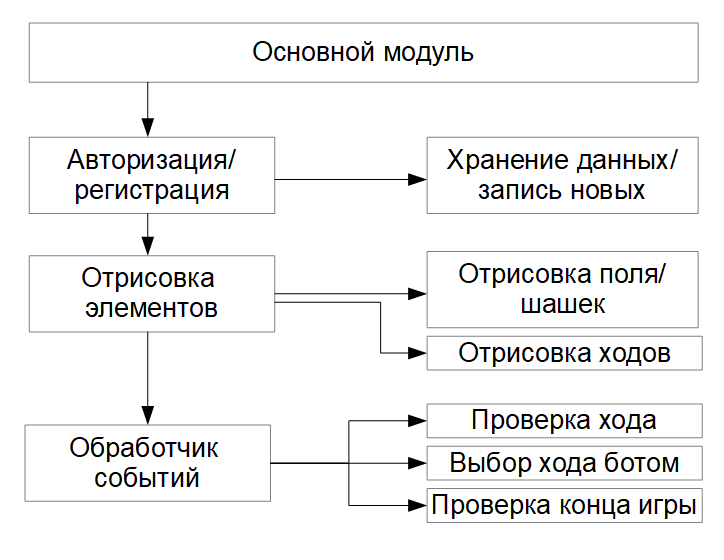
Определяется заданием на курсовую работу. Детализируется в разработанном техническом задании.

**1.2 Математические методы**

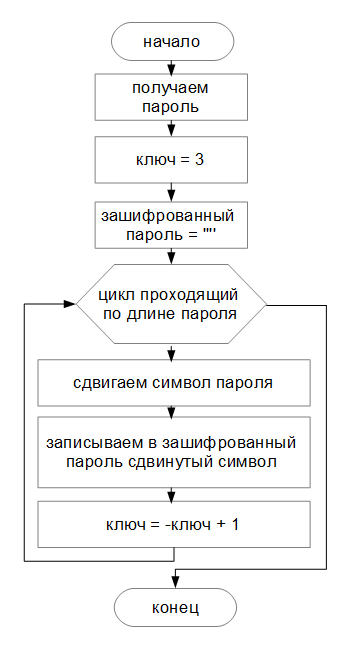
В качестве математической модели для представления и отрисовки поля был выбран двумерный массив, он позволяет легко записать положение всех шашек, выполнять все необходимые проверки и отрисовывать шашки, а также может быть легко изменен, что упрощает вывод хода пользователя. В массиве пустые клетки представлены числом 0, белые шашки 1, красные шашки 2.

**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура

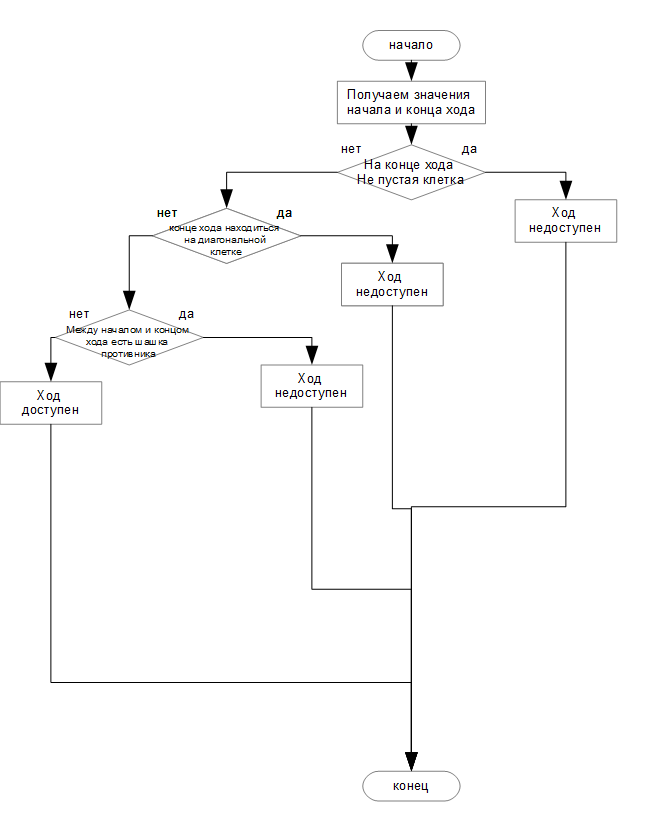


1.3.2.1. Алгоритм проверки регистрации



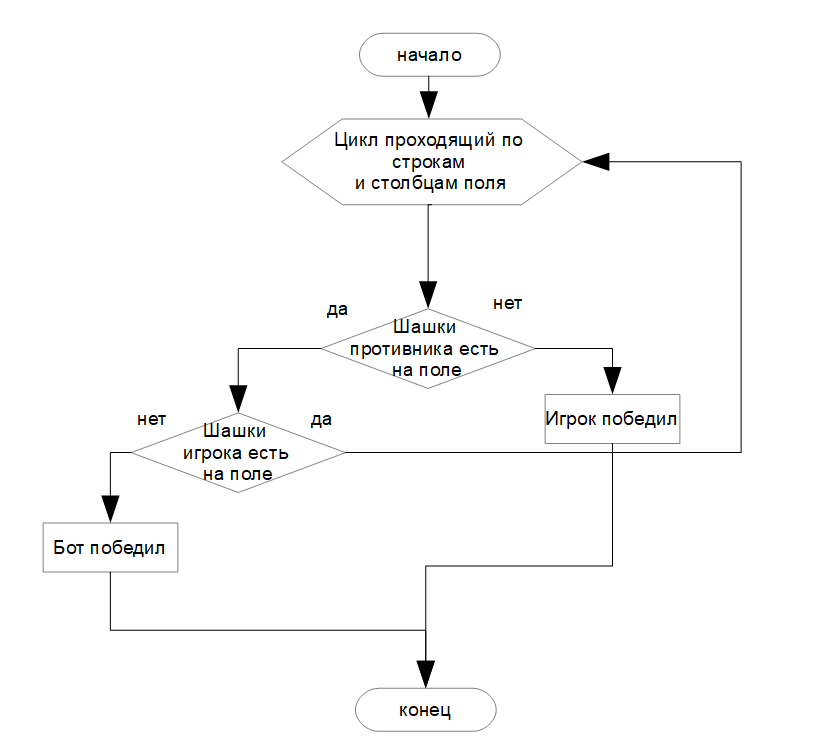
Данный алгоритм необходим для обеспечения безопасности данных пользователей. Алгоритм принимает на вход пароль пользователя, шифрует его с помощью ключа и возвращает зашифрованный пароль.

1.3.2.3. Алгоритм проверки доступности хода



Данный алгоритм необходим для контроля за соблюдением пользователем правил игры. Алгоритм проверяет возможен ли ход, который выбрал пользователь. Алгоритм получает на вход два индекса двумерного массива. Первый индекс указывает на шашку, которой пользователь хочет сходить, а второй на клетку, в которую он хочет сходить. Алгоритм проверяет возможность такого хода и, если он возможен двигает шашку пользователя, иначе ничего не делает и пользователь должен заново выбрать поле для хода.

1.3.2.4. Алгоритм проверки конца игры



Данный алгоритм необходим для того, чтобы определить победителя и корректно закончить игру. Алгоритм проверяет наличие шашек в случае их отсутствия запускает сценарий конца игры, иначе ничего не делает и игра продолжается.

**1.4 Тестирование**

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

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

**2. Источники, использованные при разработке**

1. Шишкин, В.В. Разработка логических компьютерных игр с графическим интерфейсом в среде Python [Электронный ресурс] / В.В. Шишкин, Д.С. Афонин. – Ульяновск: УлГТУ, 2023. – 89 с. – Режим доступа: для всех пользователей. – URL: <http://lib.ulstu.ru/venec/disk/2023/112.pdf> (дата обращения: 22.10.23).

2. Руководство по программированию на Tkinter и Python [Электронный ресурс]. – Режим доступа: для всех пользователей. – URL: <https://metanit.com/python/tkinter/?ysclid=lq9y8691w1850842283> (дата обращения: 26.10.23

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
 ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

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

# **Руководство программиста**

**Тема:** Компьютерная логическая игра «Апит Содок»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/3457-49 РП-03

Листов 14

**Исполнитель**:

студент гр. ИСТбд-21

*Орешин Дмитрий Юрьевич*

« » 2023 г.

**2023**

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложение «Апит-Содок» реализовано для развлечения пользователей. Приложение реализовано по следующим правилам:

**Поле и игроки.** Игра ведется между двумя соперниками (пользователь-компьютер) шашками разного цвета на квадратном поле размером 8х8 клеток. В начальной позиции у каждого игрока 16 шашек, построенных в первом и третьем ряду от игрока напротив шашек противника.

**Порядок ходов.** Игроки ходят по очереди. Первый ход делает пользователь или компьютер (в начале игры даётся выбор). Во время хода игрок может передвинуть только одну свою шашку по горизонтали или вертикали, как ладью в шахматах (то есть на любое количество клеток, если на ее пути нет фигур). Ход возможен во всех направлениях.

**Взятие шашек.** Возможны два типа взятия шашек противника: срубить шашку, взяв ее в клещи (окружить шашку противника двумя своими сверху и снизу или справа и слева) своими двумя шашками, или срубить сразу две шашки противника, встав между ними одной своей. Срубленные шашки сразу удаляются с доски.

**Цель игры.** Цель игры - срубить все шашки противника. Игра заканчивается, когда у одного из игроков не остается шашек. Игрок, у которого остались шашки, считается победителем, а второй у которого их нет - проигравшим.

В приложении предоставляется возможность зарегистрироваться чтобы играть со своим аккаунтом, сыграть в логическую игру Апит Содок, в приложении реализованы все правила этой игры: порядок хода игроков, ходы шашек, взятие шашки, проверка конца игры и вывод победителя.

**1.2 Условия, необходимые для использования приложения**

Рекомендуется к использованию на Windows 10.

При создании программы использовались встроенная библиотека “os”, “string”, “random” и сторонняя библиотека “tkinter 8.6.”.

Разработка велась в “PyCharm community edition 2022.3” на версии языка программирования Python 3.10.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Значимых строк кода 363

Структура данных одна – массив.

Использованные библиотеки:

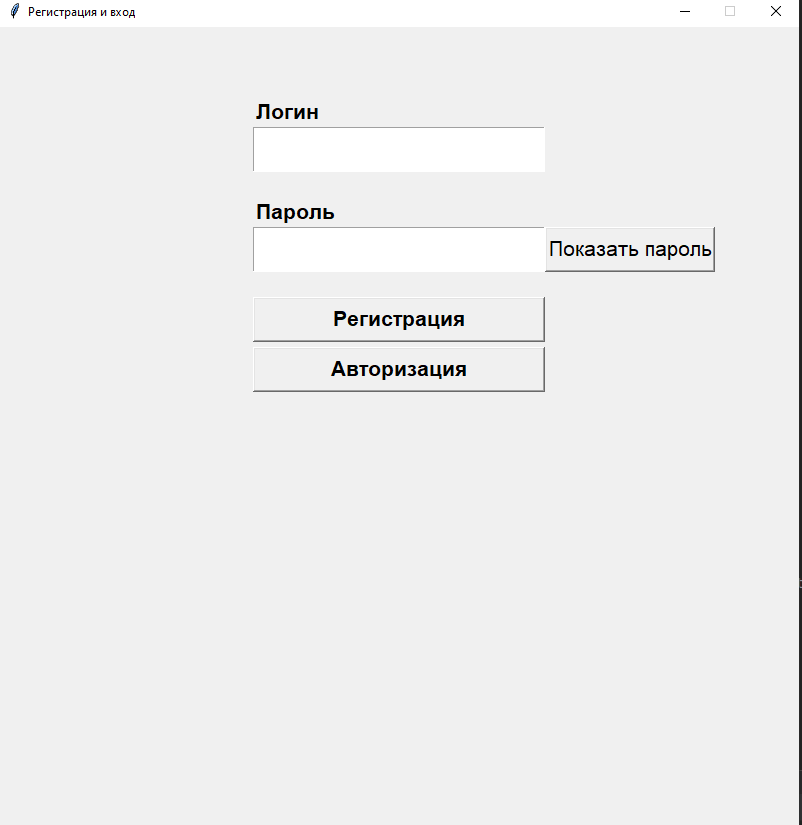
“os” - библиотека функций для работы с операционной системой. Методы, включенные в неё, позволяют определять тип операционной системы, получать доступ к переменным окружения, управлять директориями и файлами

‘’string’’ -  является частью стандартного модуля и предоставляет различные операции для работы со строками.

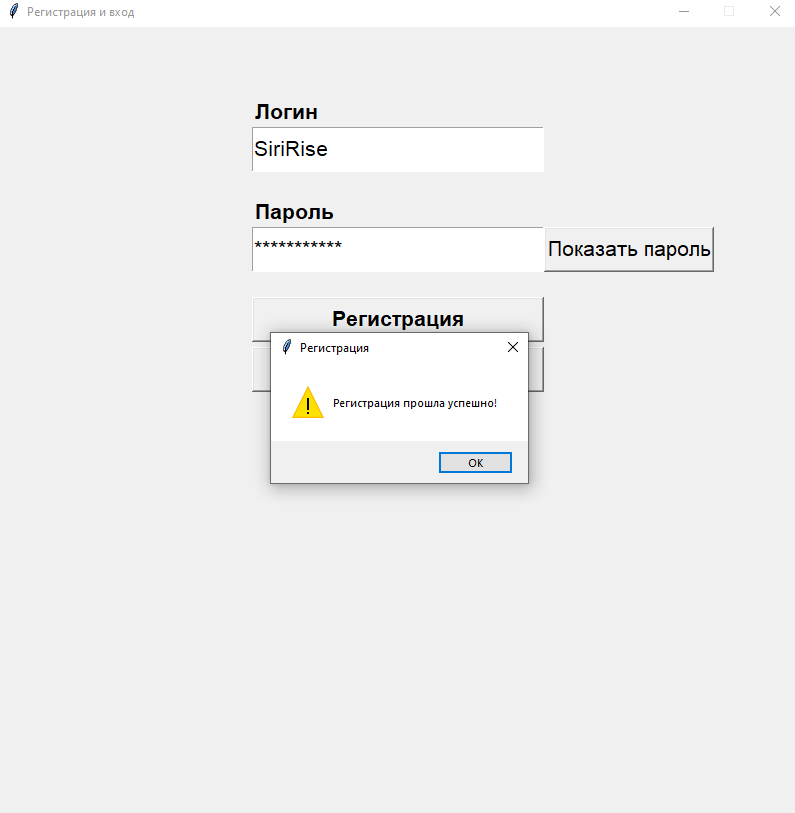
“random” - используется для генерации случайных чисел и других случайных значений. Она предоставляет функции для генерации случайных целых чисел, чисел с плавающей точкой.

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

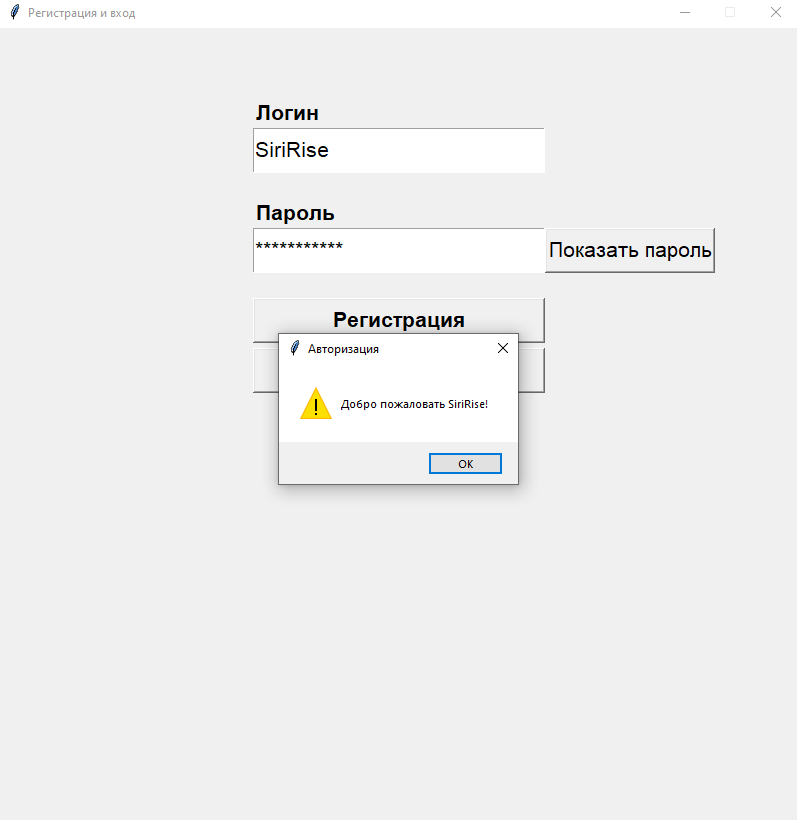
Работа приложения начинается с окна авторизации и регистрации:



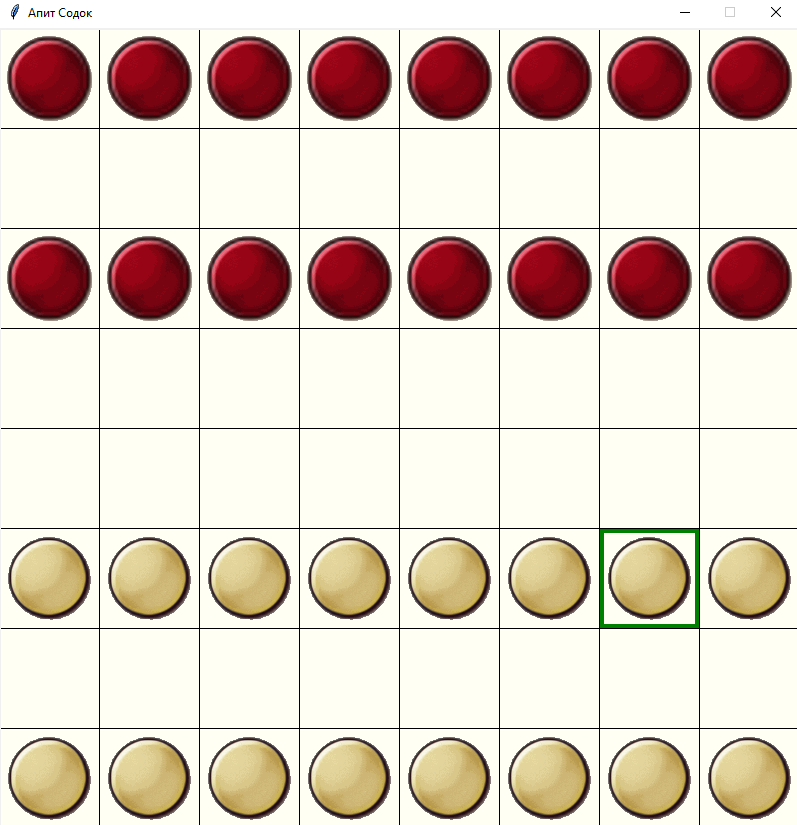
Успешная регистрация:



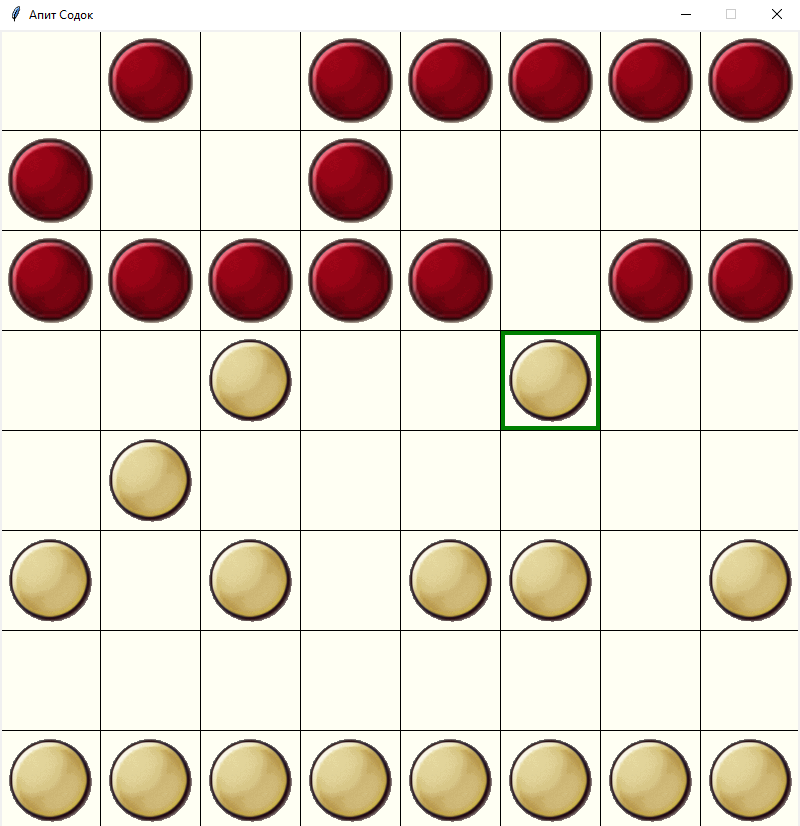
Успешная авторизация:

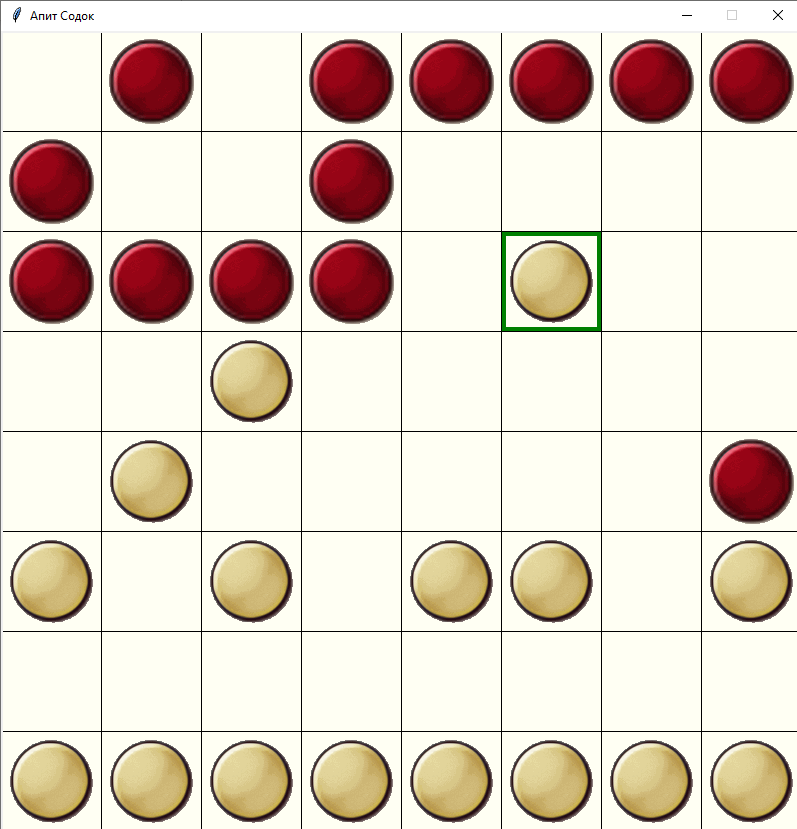


Отрисовка поля:

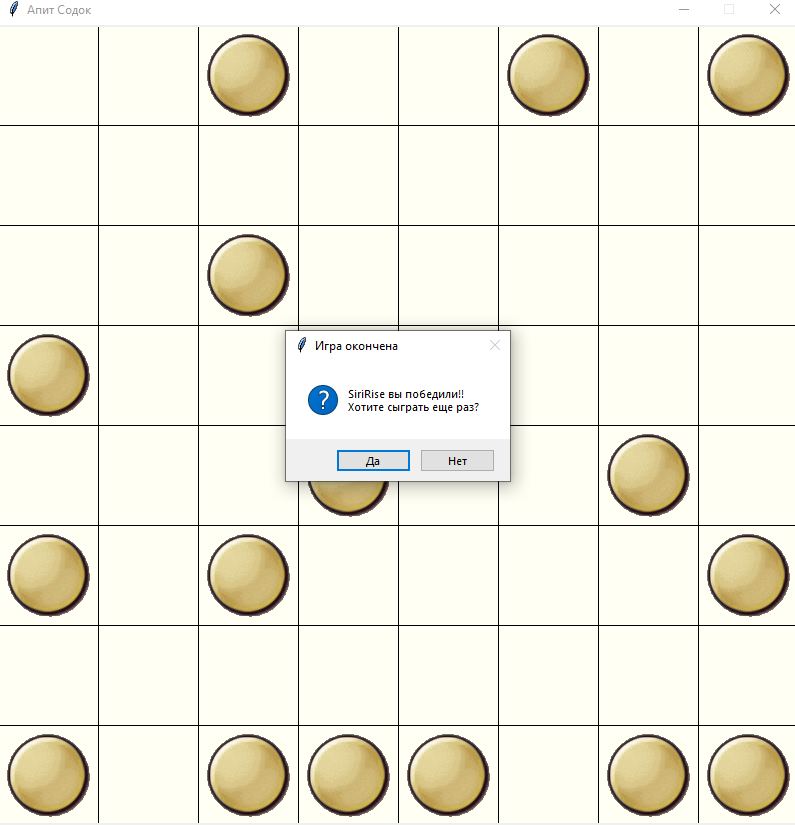


Взятие шашки:





Окончание игры и возможность перезапуска:



**2.2 Особенности реализации приложения**

В программе используется структура данных массив, эта структура была выбрана ввиду удобной навигации для отрисовки и проверок хода, взятия шашек и последовательного боя, т.к. двумерном массиве есть возможность поиска по строке и столбцу, что приближено к реальному шашечному полю, у которого также есть строки и столбцы.

**3. Обращение к программе**

1)Метод password\_encryption

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

2)Метод registration

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

3) Метод login

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

4)Метод show\_password

Скрывает пароль за определёнными символами.

5)Метод program\_start

Запускает окно с игрой.

6)Метод novaya\_igra

Запускает новую игру и сбрасывая все значения.

7)Метод vivod

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

8)Метод pozici\_1

Узнает положение курсора и узнает координаты клетки которую надо выделить зеленым.

8)Метод pozici\_2

Узнает положение курсора и узнает координаты клетки которую надо выделить зеленым. Когда клетка выделена а курсор в другом месте то клетка выделяется красным, а клетка над которой курсор выделяется зеленым.

9)Метод hod\_igroka

Ход игрока если он возможен.

10)Метод check\_available\_moves

Проверяются все возможное ходы бота и добавляются в список.

11)Метод hod\_bota

Случайно выбирает ходы из списка возможных ходов

12)Метод check\_move\_available

Проверка ходов. Проверяет нет ли на пути шашки препятствий и не ходит ли игрок по горизонтали.

13) srubit\_fishku

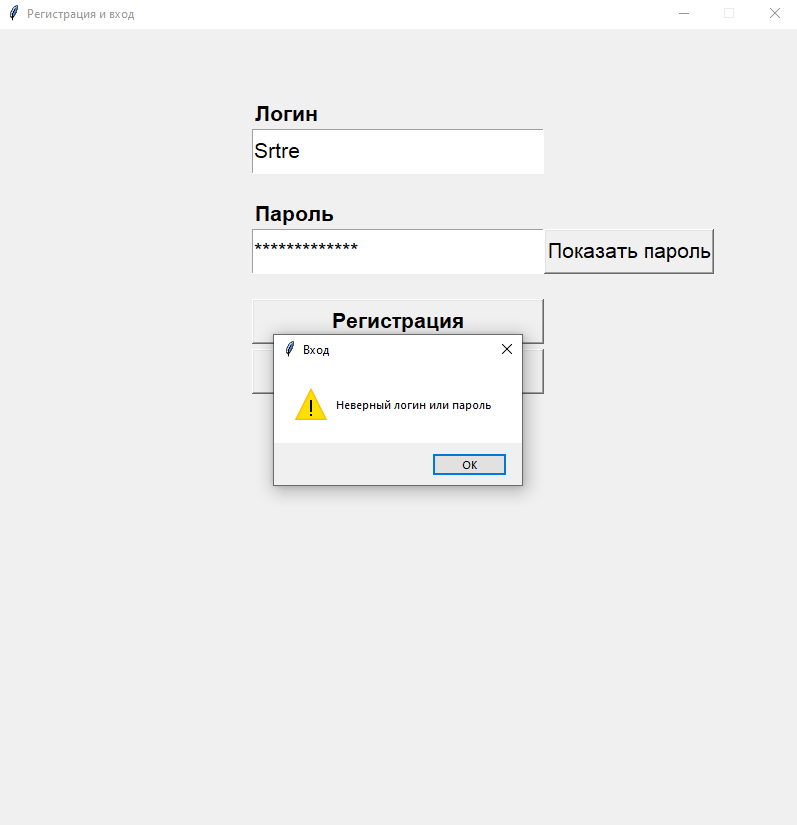
Проверяет возможность “срубания” шашек проходя по всей доске и сравнивая с условиями “срубания” шашек.

14)Метод igra\_okonchena

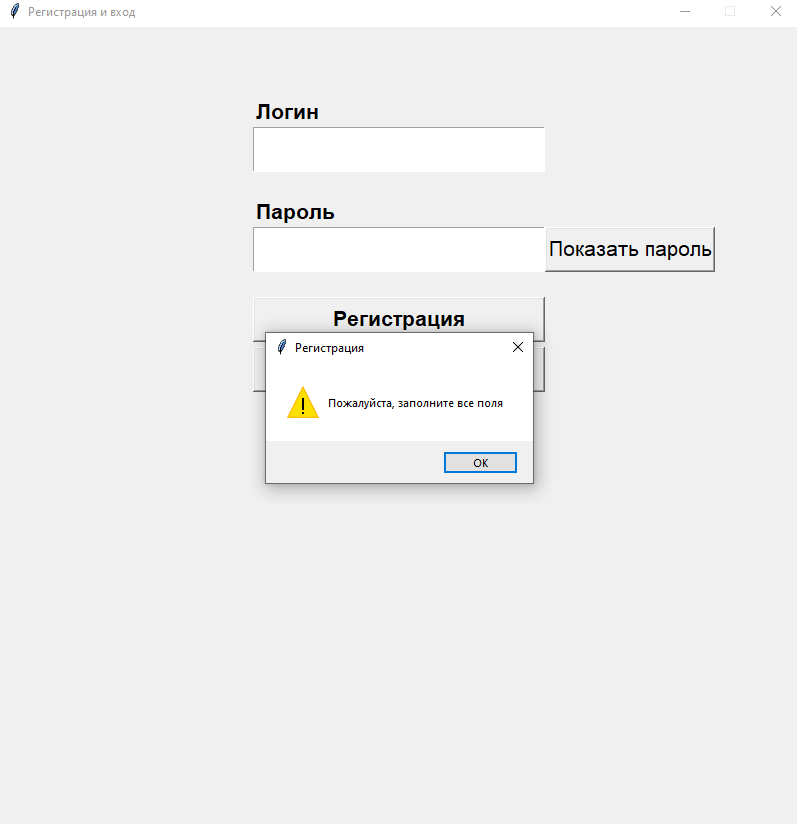
Проходит по всей доске и проверяет нет ли шашек одного из игроков. Если из нет то один из них выигрывает и предлагается начать новую игру.

**4. Сообщения**

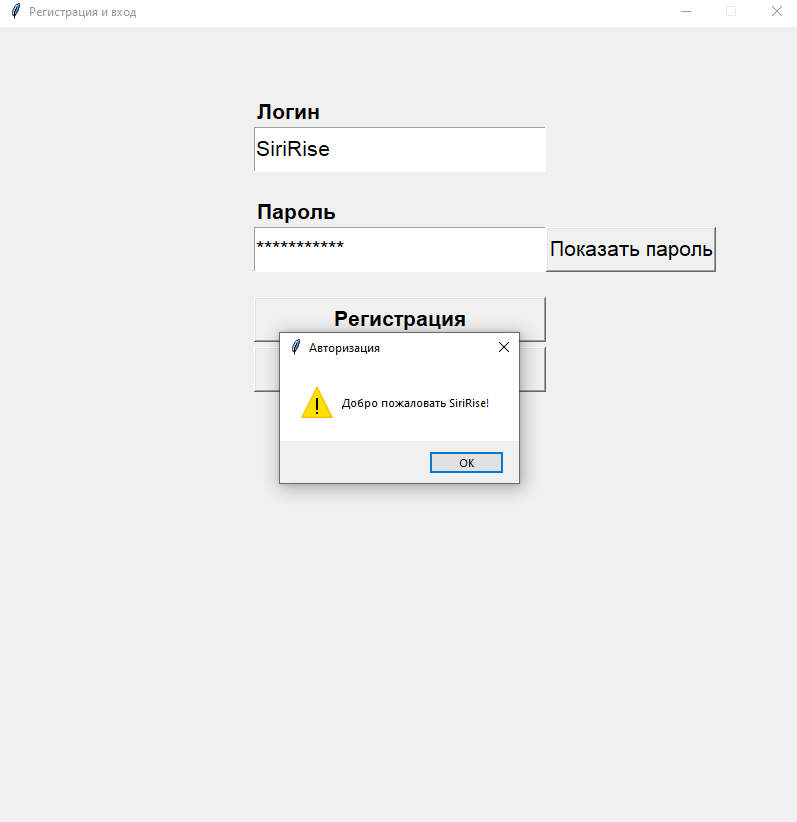
Пользователь не найден или введен неверный пароль:



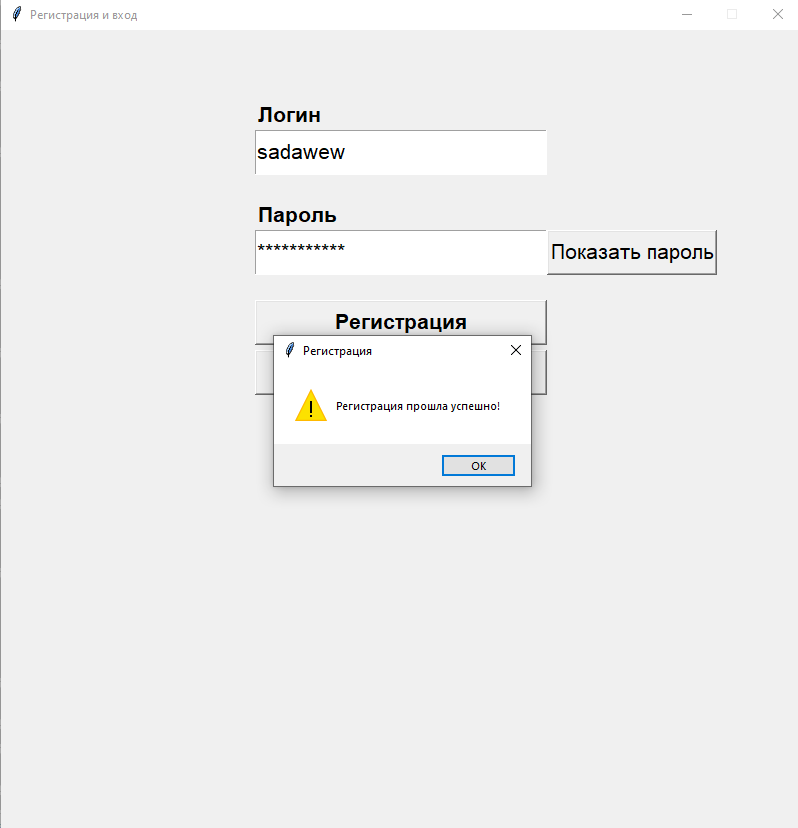
Поля для ввода пустые:



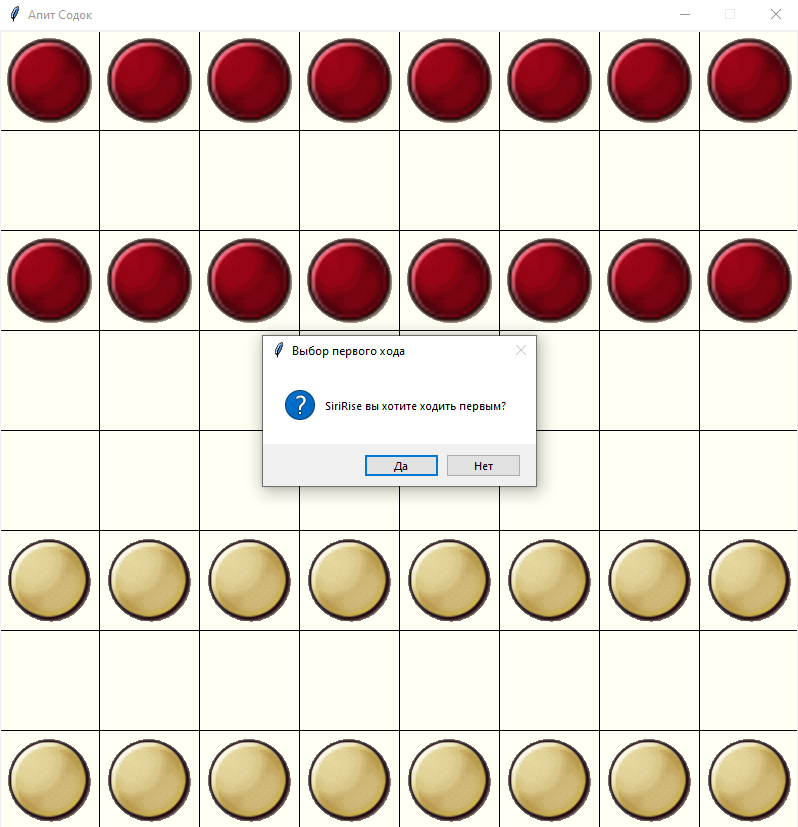
Успешная авторизация:



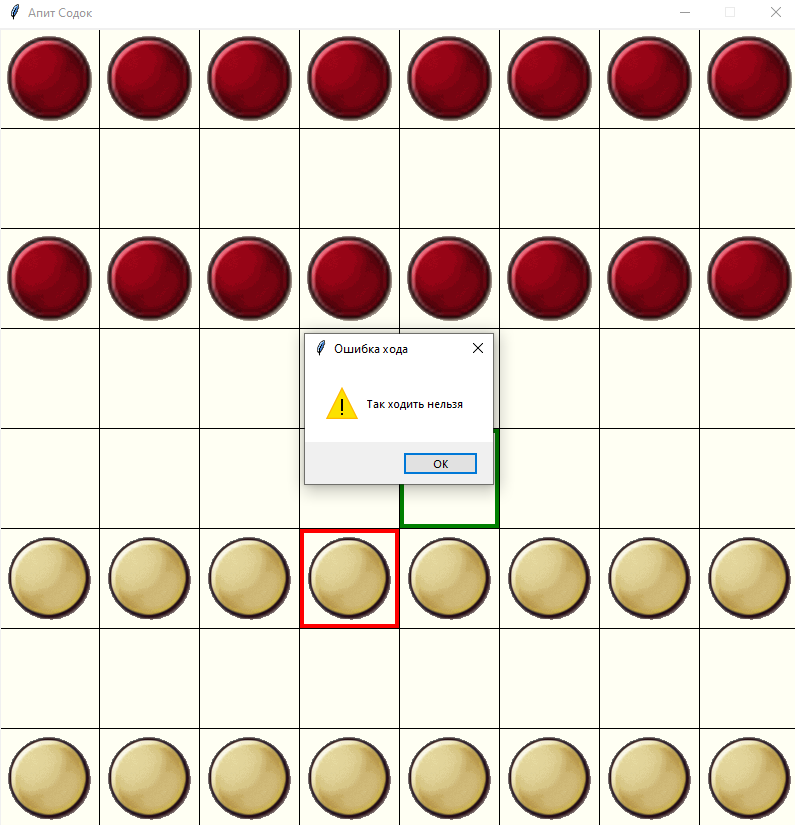
Успешная регистрация:

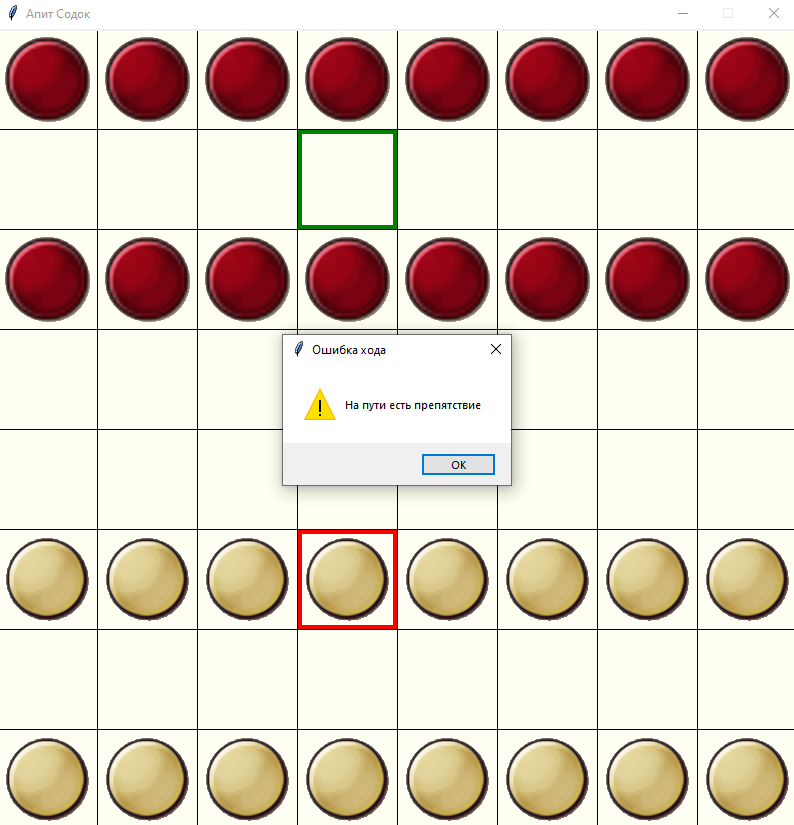


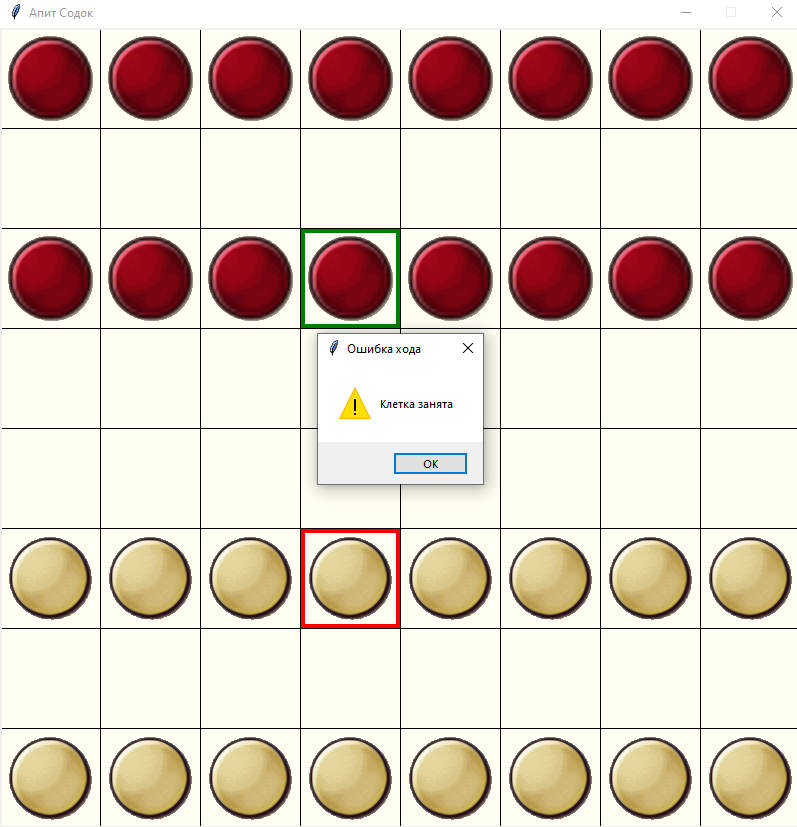
Выбор того кто первый ходит:



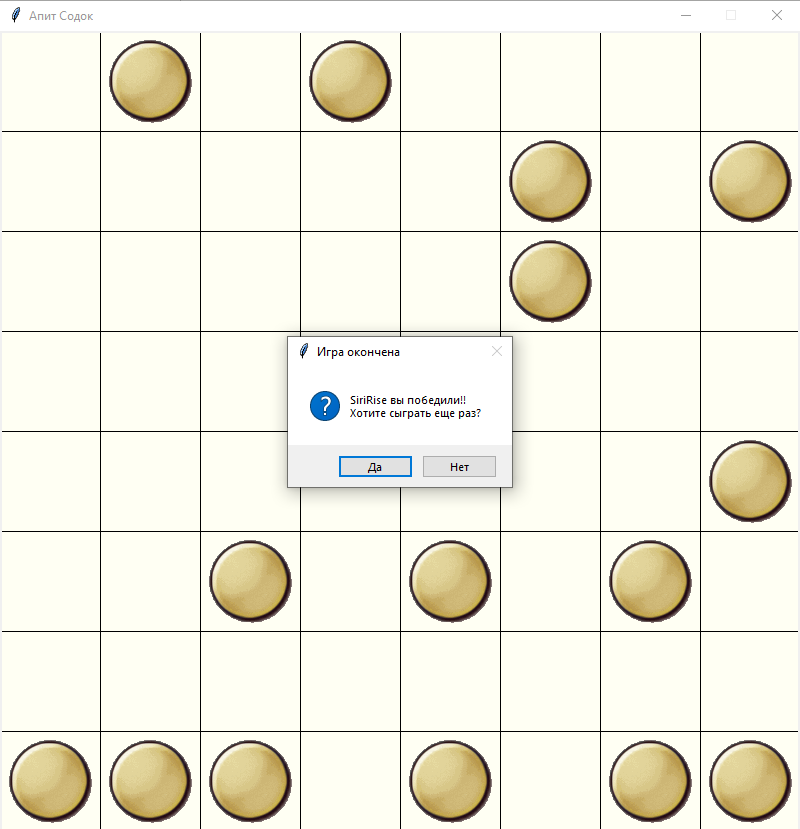
Ход неверны:







Конец игры:



# **Текст программы:**

from tkinter import \*  
import os  
from tkinter.messagebox import showwarning  
import string  
import tkinter as tk  
import random  
  
  
show\_setting = "\*"  
font\_style = ("Arial", 16, "bold")  
font\_style\_text = ("Arial", 16)  
  
  
def program\_start(username):  
 *# создаём окно* root.title('Апит Содок') *# заголовок окна* root.resizable(False, False)  
 doska = Canvas(width=800, height=800, bg='#FFFFF3')  
  
 doska.pack()  
  
 i1 = PhotoImage(file="res\\2b.gif")  
 i2 = PhotoImage(file="res\\2h.gif")  
 peshki = [0, i1, i2]  
  
 *# начинаем новую игру* def novaya\_igra():  
 global pole, current\_player, comp, player  
 pole = [[2] \* 8, [0] \* 8, [2] \* 8, [0] \* 8, [0] \* 8, [1] \* 8, [0] \* 8, [1] \* 8]  
  
 vivod(0, 0)  
 player = 1  
 comp = 2  
 current\_player = 0  
 MsgBox = tk.messagebox.askquestion("Выбор первого хода", f"{username} вы хотите ходить первым?")  
 if MsgBox == "yes":  
 current\_player = player  
 else:  
 current\_player = comp  
 root.after(500, hod\_bota)  
  
  
 def vivod(x\_poz\_1, y\_poz\_1):  
 global pole, kr\_ramka, zel\_ramka  
 cell\_size = 100  
 doska.delete('all')  
  
 *# Рисуем доску* for x in range(8):  
 for y in range(8):  
 doska.create\_rectangle(x \* cell\_size, y \* cell\_size, (x + 1) \* cell\_size, (y + 1) \* cell\_size, fill="#FFFFF3")  
  
 kr\_ramka = doska.create\_rectangle(-5, -5, -5, -5, outline="red", width=4)  
  
 zel\_ramka = doska.create\_rectangle(-5, -5, -5, -5, outline="green", width=4)  
  
 *# Рисуем стоячие пешки* for x in range(8):  
 for y in range(8):  
 z = pole[y][x]  
 if z and (x\_poz\_1, y\_poz\_1) != (x, y):  
 doska.create\_image(x \* cell\_size, y \* cell\_size, anchor=NW, image=peshki[z])  
  
 *# Рисуем активную пешку* z = pole[y\_poz\_1][x\_poz\_1]  
 if z:  
 doska.create\_image(x\_poz\_1 \* cell\_size, y\_poz\_1 \* cell\_size, anchor=NW, image=peshki[z], tag='ani')  
 doska.update() *# Обновление  
  
 # выбор клетки для хода 1* def pozici\_1(event):  
  
 x, y = (event.x) // 100, (event.y) // 100 *# вычисляем координаты клетки* if 0 <= x < 8 and 0 <= y < 8:  
 doska.coords(zel\_ramka, x \* 100 + 3, y \* 100 + 3, x \* 100 + 98, y \* 100 + 98)  
  
  
 def pozici\_2(event):  
 global poz1\_x, poz1\_y, poz2\_x, poz2\_y  
 try:  
 x, y = (event.x) // 100, (event.y) // 100  
 if pole[y][x] == player:  
 doska.coords(kr\_ramka, x \* 100 + 3, y \* 100 + 3, x \* 100 + 98, y \* 100 + 98)  
 poz1\_x, poz1\_y = x, y  
 elif pole[poz1\_y][poz1\_x] == player:  
 poz2\_x, poz2\_y = x, y  
 if check\_move\_available(poz1\_x, poz1\_y, poz2\_x, poz2\_y):  
 hod\_igroka()  
  
 doska.coords(kr\_ramka, -5, -5, -5, -5)  
 except tk.TclError:  
 pass  
 except NameError:  
 pass  
  
  
 def hod\_igroka(): *# Ход игрока* global poz1\_x, poz1\_y, poz2\_x, poz2\_y, player , current\_player  
  
 if pole[poz1\_y][poz1\_x] == player and check\_move\_available(poz1\_x, poz1\_y, poz2\_x, poz2\_y):  
 *# делаем ход* pole[poz2\_y][poz2\_x], pole[poz1\_y][poz1\_x] = pole[poz1\_y][poz1\_x], pole[poz2\_y][poz2\_x]  
 srubit\_fishku(poz2\_x, poz2\_y)  
 vivod(0, 0)  
  
 *# передаем ход другому игроку* if igra\_okonchena():  
 return True  
 else:  
 current\_player = comp  
 root.after(500, hod\_bota)  
  
  
 def check\_available\_moves(x, y, dx, dy):  
 moves = []  
 new\_x, new\_y = x + dx, y + dy  
 while 0 <= new\_x < 8 and 0 <= new\_y < 8:  
 if pole[new\_y][new\_x] == 0:  
 moves.append((x, y, new\_x, new\_y))  
 new\_x += dx  
 new\_y += dy  
 else:  
 break  
 return moves  
  
  
 def hod\_bota():  
 global pole, current\_player  
  
 available\_moves = [] *# Список доступных ходов для каждой пешки* for y, row in enumerate(pole):  
 for x, cell in enumerate(row):  
 if cell == comp: *# Если находим пешку противника, проверяем возможные ходы* available\_moves.extend(check\_available\_moves(x, y, -1, 0)) *# Проверяем ходы влево* available\_moves.extend(check\_available\_moves(x, y, 1, 0)) *# Проверяем ходы вправо* available\_moves.extend(check\_available\_moves(x, y, 0, -1)) *# Проверяем ходы вверх* available\_moves.extend(check\_available\_moves(x, y, 0, 1)) *# Проверяем ходы вниз* if available\_moves: *# Если есть доступные ходы* move = random.choice(available\_moves) *# Выбираем случайный ход* x\_start, y\_start, x\_end, y\_end = move  
 *# Совершаем ход* pole[y\_end][x\_end], pole[y\_start][x\_start] = pole[y\_start][x\_start], 0  
 srubit\_fishku(x\_end, y\_end)  
 vivod(0, 0)  
 if igra\_okonchena():  
 return True  
 else:  
 current\_player = player  
 else:  
 MsgBox = tk.messagebox.askquestion("Игра окончена", f"{username} вы победили!!\n" "У противника нет доступных ходов\n" "Хотите сыграть еще раз?")  
 if MsgBox == "yes":  
 vivod(0, 0)  
 novaya\_igra()  
 else:  
 root.destroy()  
 return True  
  
  
 def check\_move\_available(x1, y1, x2, y2): *# Проверка ходов* global pole  
 *# проверить, нет ли препятствий (финальная клетка пустая)* if pole[y2][x2] != 0:  
 showwarning("Ошибка хода", "Клетка занята")  
 return False  
  
 *# шаг должен быть впереди или назад, влево или вправо* if x1 != x2 and x1 != -1 and y1 != y2:  
 showwarning("Ошибка хода", "Так ходить нельзя")  
 return False *# был сделан ни горизонтальный, ни вертикальный ход* if x1 == x2: *# проверка хода по вертикали* step = 1 if y1 < y2 else -1  
 for y in range(y1 + step, y2, step):  
 if pole[y][x1] != 0:  
 showwarning("Ошибка хода", "На пути есть препятствие")  
 return False *# если есть препятствие, то ход невозможен* else: *# проверка хода по горизонтали* step = 1 if x1 < x2 else -1  
 for x in range(x1 + step, x2, step):  
 if pole[y1][x] != 0:  
 showwarning("Ошибка хода", "На пути есть препятствие")  
 return False *# если есть препятствие, то ход невозможен* return True  
  
  
 def srubit\_fishku(x, y): *# Функция "срубания" фигур* global pole  
  
 for y in range(8):  
 for x in range(8):  
 if pole[y][x] == current\_player and pole[y][x] != 0:  
 if 0 < y < len(pole) - 1 and pole[y - 1][x] != current\_player and pole[y - 1][x] != 0 and pole[y + 1][x] != current\_player and pole[y + 1][x] != 0:  
 pole[y - 1][x] = pole[y + 1][x] = 0  
 elif 0 < x < len(pole[0]) - 1 and pole[y][x - 1] != current\_player and pole[y][x - 1] != 0 and pole[y][x + 1] != current\_player and pole[y][x + 1] != 0:  
 pole[y][x - 1] = pole[y][x + 1] = 0  
 elif pole[y][x] != current\_player and pole[y][x] != 0:  
 if 0 < y < len(pole) - 1 and pole[y - 1][x] == current\_player and pole[y - 1][x] != 0 and pole[y + 1][x] == current\_player and pole[y + 1][x] != 0:  
 pole[y][x] = 0  
 elif 0 < x < len(pole[0]) - 1 and pole[y][x - 1] == current\_player and pole[y][x - 1] != 0 and pole[y][x + 1] == current\_player and pole[y][x + 1] != 0:  
 pole[y][x] = 0  
  
  
 def igra\_okonchena():  
 *# Проверка, есть ли на поле фигурки для каждого игрока* player\_remaining = any(player in row for row in pole)  
 comp\_remaining = any(comp in row for row in pole)  
  
 *# Если одному из игроков нечем ходить, игра окончена* if not comp\_remaining:  
 MsgBox = tk.messagebox.askquestion("Игра окончена", f"{username} вы победили!!\n""Хотите сыграть еще раз?")  
 if MsgBox == "yes":  
 vivod(0, 0)  
 novaya\_igra()  
 else:  
 root.destroy()  
 return True  
 elif not player\_remaining:  
 MsgBox = tk.messagebox.askquestion("Игра окончена", f"{username} вы проиграли!!\n""Хотите сыграть еще раз?")  
 if MsgBox == "yes":  
 vivod(0, 0)  
 novaya\_igra()  
 else:  
 root.destroy()  
 return True  
  
  
 novaya\_igra()  
 vivod(0, 0) *# рисуем игровое поле* doska.bind("<Motion>", pozici\_1) *# движение мышки по полю* doska.bind("<Button-1>", pozici\_2) *# нажатие левой кнопки*def password\_encryption(password):  
 key = 3  
 encryption\_password = ""  
 for i in password:  
 encryption\_password\_temp = chr(ord(i) + key)  
 encryption\_password += encryption\_password\_temp  
 key = -key + 1  
 return encryption\_password  
  
  
def registration():  
 username = entry\_username.get()  
 password\_raw = entry\_password.get()  
 password = password\_encryption(password\_raw)  
  
 if not username or not password:  
 showwarning("Регистрация", "Пожалуйста, заполните все поля")  
 return  
 if len(username) <= 2:  
 showwarning("Регистрация", "Имя пользователя должно быть длиннее 2 символов")  
 return  
 if len(password) < 8 or not any(char in password for char in string.punctuation):  
 showwarning("Регистрация", "Пароль должен быть длиннее 7 символов и содержать хотя бы один специальный символ")  
 return  
  
 print(username, password)  
 with open("user\_data.txt", "r") as file:  
 lines = file.readlines()  
 for line in lines:  
 stored\_username, \_ = line.strip().split('•')  
 if username == stored\_username:  
 showwarning("Регистрация", "Пользователь с таким именем уже существует")  
 return  
  
 with open("user\_data.txt", "a") as file:  
 file.write(f'{username}•{password}\n')  
 showwarning("Регистрация", "Регистрация прошла успешно!")  
 program\_start(username)  
  
  
def login():  
 username = entry\_username.get()  
 password\_raw = entry\_password.get()  
 password = password\_encryption(password\_raw)  
  
 if not username or not password:  
 showwarning("Авторизация", "Пожалуйста, заполните все поля")  
 return  
  
 with open("user\_data.txt", "r") as file:  
 lines = file.readlines()  
 user\_found = False  
 incorrect\_password = False  
  
 for line in lines:  
 stored\_username, stored\_password = line.strip().split('•')  
 if username == stored\_username:  
 if password == stored\_password:  
 user\_found = True  
 showwarning("Авторизация", f"Добро пожаловать {username}!")  
 program\_start(username)  
 break  
 else:  
 incorrect\_password = True  
 break  
 if not user\_found or incorrect\_password:  
 showwarning("Вход", "Неверный логин или пароль")  
 return  
  
  
if not os.path.exists("user\_data.txt"):  
 with open("user\_data.txt", "w"):  
 pass  
  
  
def show\_password():  
 global show\_setting  
 if show\_setting == "\*":  
 show\_setting = ""  
 button\_text = "Скрыть пароль"  
 else:  
 show\_setting = "\*"  
 button\_text = "Показать пароль"  
 try:  
 button\_show\_password.config(text=button\_text)  
 entry\_password.config(show=show\_setting)  
 except (NameError):  
 pass  
  
  
root = Tk()  
root.resizable(False, False)  
root.title("Регистрация и вход")  
width = 800  
height = 800  
  
screen\_width = root.winfo\_screenwidth()  
screen\_height = root.winfo\_screenheight()  
  
x = (screen\_width/2) - (width/2)  
y = (screen\_height/2) - (height/2)  
  
root.geometry("%dx%d+%d+%d" % (width, height, x, y))  
  
label\_username = Label(root, text="Логин", font=font\_style)  
label\_username.place(x=254, y=70)  
  
entry\_username = Entry(root, font=font\_style\_text)  
entry\_username.place(x=254, y=100, width=292, height=45)  
  
label\_password = Label(root, text="Пароль", font=font\_style)  
label\_password.place(x=254, y=170)  
  
entry\_password = Entry(root, show=show\_setting, font=font\_style\_text)  
entry\_password.place(x=254, y=200, width=292, height=45)  
  
button\_reg = Button(root, text="Регистрация", command=registration, font=font\_style)  
button\_reg.place(x=254, y=270, width=292, height=45)  
  
button\_log = Button(root, text="Авторизация", command=login, font=font\_style)  
button\_log.place(x=254, y=320, width=292, height=45)  
  
button\_show\_password = Button(root, text="Показать пароль", command=show\_password, font=font\_style\_text)  
button\_show\_password.place(x=546, y=200, width = 170, height=45)  
  
mainloop()