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«Го-Бан»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337. 22/329-57 ТЗ-01

Листов: 43

Руководитель разработки:

к. т. н., доцент

Шишкин Вадим Викторинович

« » 202\_\_ г.

Исполнитель:

студент гр. ИСТбд-32

Быкин Вадим

Дмитриевич

« » 202\_\_ г.

202\_\_ г.

Содержание

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

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

Пояснительная записка...…………………………………………….......10

Руководство программиста……………………………….………….......22

Текст программы…..………………………………………………….......30

**Аннотация**

Данный документ представляет собой пояснительную записку на курсовую работу на тему «Компьютерная логическая игра ”Го-Бан”». 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

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

по дисциплине «Алгоритмы и структуры данных»

Тема: Компьютерная логическая игра «Го-Бан»

Р.02069337. 22/2359-27 ТЗ-01

Листов: 6

**Исполнитель**:

студент гр. ИСТбд-32

Быкин Вадим Дмитриевич

« » 202\_\_ г.

202\_\_ г.

**Введение**

Курсовая работа представляет собой однопользовательское десктопное приложение, компьютерную игру «Го-Бан»

Краткие правила игры:

* Игра ведётся на поле 8х8 в два этапа. У игроков по 12 камней.
* В первом этапе игры игроки по очереди выставляют свои камни на поле (по одному за ход).
* В первом этапе игры существует возможность выставить пять своих камней в ряд по горизонтали, вертикали и диагонали.
* Во втором этапе, игроки передвигают свои камни.
* Атаковать или перепрыгивать через камни противника нельзя.
* Игра заканчивается, когда у какого-либо игрока собрана хотя бы один ряд из пяти камней.

Функциональные возможности:

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

**1. Основания для разработки**

В качестве основания для разработки указывается учебный план направления 09.03.02 «Информационные системы и технологии».

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

**2.1. Функциональное назначение**

Требуется разработать однопользовательское десктопное приложение по игре в «Го-Бан» с графическим интерфейсом в среде Windows.

**2.2. Требования к функциональным характеристикам**

Приложение должно соответствовать следующим правилам игры:

Игра ведётся между двумя соперниками (пользователь-компьютер) камнями разного цвета на квадратном поле размером 8х8 клеток. Для доски 8х8 каждому игроку выдаётся 12 шашек или камней в китайской терминологии. Цель игры – поставить пять камней в один ряд, по вертикали, горизонтали или диагонали. Игра состоит из двух этапов:

**Первый этап.** Игроки совершают ходы поочерёдно. Первый ход делает пользователь. В процессе хода игрок может переместить только один свой камень. Игроки выставляют свои камни на доску, стараясь поставить пятёрку камней уже на этом этапе. Реальная цель этого этапа не поставить пять камней, а захватить возможно большее пространство для манёвра.

**Второй этап.** Игроки начинают двигать камни с той же целью – поставить пять камней в ряд. Пятёркой же считается: вертикаль, горизонталь или диагональ. Но двигать камни можно только по вертикали или горизонтали. За один ход можно передвинуть только один камень на одну клетку. Перепрыгивать через камни нельзя. Возможности срубить камень противника нет. Все камни находятся на доске в течение всей игры. В процессе игры, на втором этапе важно ограничить подвижность камней противника и при этом обеспечить своим камням большее пространство для манёвра.

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;

- интерактивные приёмы, 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. Игровые изображения могут храниться в отдельных графических файлах. Логин и пароль пользователя должны вводиться с клавиатуры. Логины и пароли зарегистрированных пользователей должны храниться в отдельном файле или базе данных в зашифрованном виде. Пояснительные информационные сообщения для пользователя должны выводиться сбоку от игрового поля 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 сопряжения модулей (методов) программы; комплексное тестирование программы.

**2.4. Требования к информационной и программной совместимости**

Операционная система: Windows 10

Используемые библиотеки: tkinter, PIL, sqlite3, math

Язык: Python 3.9

Среда разработки: PyCharm Community Edition 2021.2.2

**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

ПРАВИЛЬНО:

1. Брать диски за внутреннее кольцо или за внешний край

2. Использовать специальный маркер для оптических дисков, не содержащий растворителя, для нанесения надписей на поверхность дисков

3. Избегать попадания грязи и других инородных предметов на поверхность дисков

4. Хранить диски в вертикальном положении (книжный способ) в пластиковых кейсах, предназначенных для CD и DVD

5. Помещать диск обратно в кейс сразу же после использования

6. Хранить диски всегда в упаковке для уменьшения влияния внешней среды

7. Открывать упаковку с дисками непосредственно перед записью

8. Хранить диски в темном, прохладном, сухом, проветриваемом помещении

9. Удалять грязь, инородные предметы, отпечатки пальцев, пятна и т.п. с поверхности дисков чистой хлопковой тканью легким движением от центра диска к внешнему краю

10. Использовать специальные чистящие средства для CD и DVD, изопропиленовый спирт или метанол для чистки поверхности дисков

11. Перед началом процесса записи проверять состояние поверхности диска

НЕПРАВИЛЬНО:

1. Прикасаться к поверхности диска

2. Гнуть диск

3. Размещать на диске наклейки

4. Хранить диски в горизонтальном положении длительное время (годами)

5. Открывать упаковку с диском задолго до начала записи

6. Хранить диски в очень жарком или очень влажном помещении

7. Хранить диски в условиях с большим колебанием температуры и влажности

8. Наносить надписи, пометки на пишущую поверхность диска (которую «читает» лазер)

9. Чистить диск круговыми движениями

ОСОБЕННО ДЛЯ CD НЕПРАВИЛЬНО:

1. Царапать нерабочую поверхность диска

2. Использовать ручки, карандаши или непредназначенные для CD фломастеры для нанесения надписей

3. Наносить надписи маркерами, содержащими растворители

4. Снимать или переклеивать наклейки

УСЛОВИЯ ХРАНЕНИЯ:

Температура – от 4°С до 20°С

Относительная влажность – от 20% до 50%

Температура 18°С и влажность 40% считаются подходящими для длительного хранения.

Более низкая температура и влажность считаются подходящими для продленного срока хранения.

2.6.3. Сроки хранения

Срок хранения – до июля 2025 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.

**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«Го-Бан»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

**Пояснительная записка**

Р.02069337. 22/2359-27 ПЗ-01

Листов: 12

**Исполнитель**:

студент гр. ИСТбд-32

Быкин Вадим Дмитриевич

« » 202\_\_ г.

202\_\_ г.

**Введение**

Программа предоставляет собой логическую игру «Го-Бан».

**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 разрабатываемом приложении в качестве модели выступает поле 10х10 для компьютера и 8х6 визуальное поле для игрока. А также модель поиска выгодной компьютеру клетки для преимущества над игроком.

Модель поля для компьютера:

["-", "-", "-", "-", "-", "-", "-", "-", "-", "-"],

["|", 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, 0, 0, 0, 0, 0, "|"],

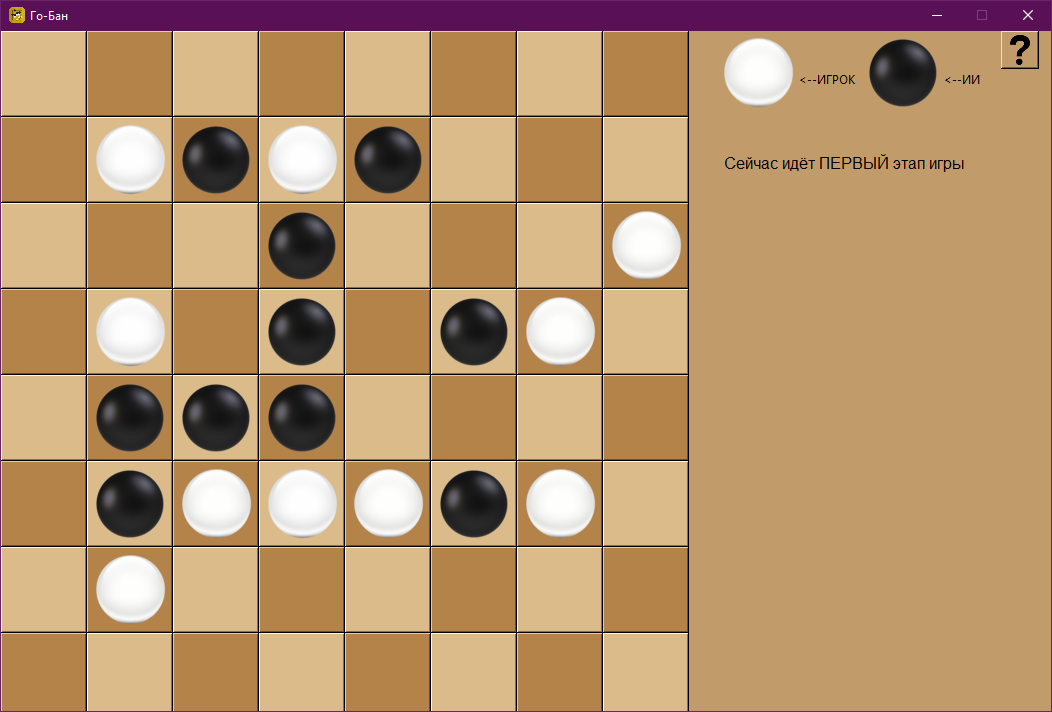
["|", 0, 0, 0, 0, 0, 0, 0, 0, "|"],

["-", "-", "-", "-", "-", "-", "-", "-", "-", "-"]

0 – (пустые) клетки свободны, 1 – клетка с фишкой игрока, 2 – клетка с фишкой компьютера, "-" и "|" – ограничение поля.

Модель поля для игрока

Первый этап:



Второй этап:



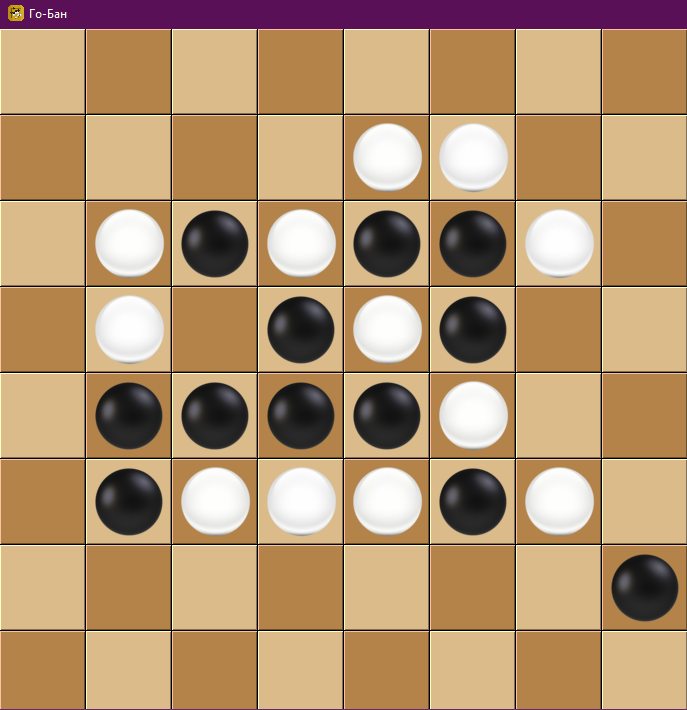
**Модель поведения компьютера в первом этапе**

Компьютер в первом этапе игры будет стараться предотвращать выставление пятёрок игроком, при этом он также будет стараться сам выставить пятёрку. Делает он это, ставя камни на продолжении двух или трёх рядом стоящих камней игрока (левый рисунок). Если таких возможностей нет или игрок не создаст подобную ситуацию, то компьютер будет стараться ставить свои фишки рядом друг с другом, чтобы у него были возможные пятёрки для второго этапа (правый рисунок). (белые камни – игрок, чёрные – компьютер).

**Модель поведения компьютера во втором этапе**

Во втором этапе игры компьютер будет стремиться собрать пять камней в ряд, если такая возможность есть. В данном примере показан возможный вариант игры, в котором компьютер собирает пять камней.

**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура

Основной модуль

с глобальными структурами данных

Регистрация/

Авторизация

Определение (отрисовка) основных виджетов

Определение конца игры

Обработчик событий

Оценка хода (оценочная функция)

Расчет и визуализация хода компьютера

1.3.2. Алгоритм создания игрового поля:

Конец игры

да

да

нет

нет

Условия конца игры выполнены? (второй этап)

Ход компьютера

Условия конца игры выполнены? (второй этап)

да

нет

Ход правильный?

Ход игрока

(1 этап – выставление,

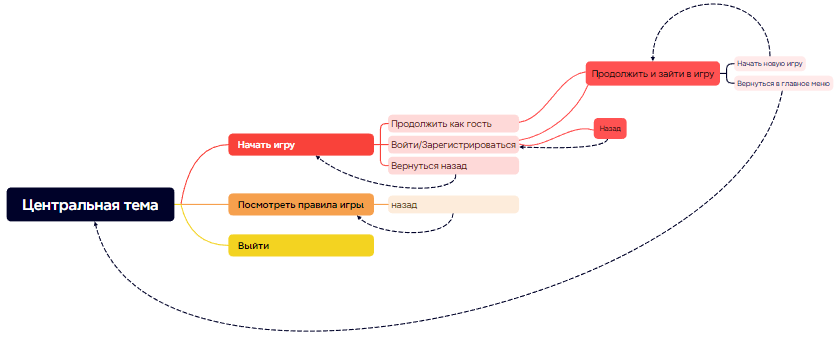
2 этап – передвижение)

Создание игрового поля

Начало

**1.4 Тестирование**

Интеллектуальная карта приложения



1.4.1 Описание отчета о тестировании

В данном отчете представлены результаты тестирования программы на основе разработанных test-case и чек-листов и статического тестирования документации и программного кода. 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 тести-

рования следует исправление выявленных багов.

1.4.3 Методика тестирования

Тестирование проводилось с использованием следующих методов:

– Статическое тестирование: анализ и проверка кода без его запуска, выявление ошибок в технической документации;

– Ручное тестирование: запуск пользовательских сценариев программы с различными входными данными и проверка корректности полученных результатов;

1.4.4 Проведенные тесты

4.1) Статическое тестирование

Количество обнаруженных и исправленных ошибок в документации: 8.

Количество обнаруженных и исправленных ошибок в программном

коде: 14.

В ходе тестирования были проведены следующие тесты:

4.2) Ручное тестирование

Написаны и проведены следующие тест-кейсы и чек-листы:

**TK1.** Проверка отрисовки игрового поля

Шаги:

1. Запустить приложение.

2. В появившемся окне нажать кнопку ‘Начать игру’, затем ‘Регистрация’.

3.Заполнить поля и нажать ‘Зарегистрироваться’

Ожидаемый результат:

Должно открыться окно с отрисованным игровым полем и виджетами.

Фактический результат:

Открылось окно с отрисованным игровым полем и виджетами.

**ТК2.** Проверка хода

1. Запустить приложение

2. В появившемся окне нажать кнопку ‘Начать игру’, затем ‘Регистрация’.

3. Заполнить поля и нажать ‘Зарегистрироваться’

4. Проверить возможность передвижения камней, а также верное выполнение проверки хода игрока.

Ожидаемый результат:

Свободное передвижение камней (в рамках правил), а также получение уведомления в случае неверного хода.

Фактический результат:

Свободное передвижение камней (в рамках правил), а также получение уведомления в случае неверного хода.

**ТК3.** Проверка окончания игры.

1. Запустить приложение.

2. В появившемся окне нажать кнопку ‘Начать игру’, затем ‘Регистрация’.

3. Заполнить поля и нажать ‘Зарегистрироваться’.

4. Дойти до стадии игры, когда выполнится условие окончание игры.

Ожидаемый результат:

Появление предупреждения об окончании игры, открытие окна с виджетами для выхода из игры или перезапуском игры.

Фактический результат:

Появление предупреждения об окончании игры, открытие окна с виджетами для выхода из игры или перезапуском игры.

**ТК4.** Проверка правильности отрисовки всех виджетов и изображений.

1. Запустить приложение.

2. В появившихся окнах поочередно нажимать все кнопки.

3. Внимательно просмотреть наличие изображений и виджетов.

Ожидаемый результат:

Корректная работа всех изображений и виджетов.

Фактический результат:

Корректная работа всех изображений и виджетов.

**ТК5**. Проверка хода компьютера.

1. Запустить приложение.

2. В появившемся окне нажать кнопку ‘Начать игру’, затем ‘Регистрация’.

3. Заполнить поля и нажать ‘Зарегистрироваться’

4. Проверить, выполняются ли правила во время хода компьютера.

Ожидаемый результат:

Ход компьютера отображается правильно. Компьютер соблюдает правила игры.

Фактический результат:

Ход компьютера отображается правильно. Компьютер соблюдает правила игры.

**ТК6.** Проверка корректности работы всех кнопок в соответствии с архитектурой.

1. Запустить приложение
2. В появившихся окнах поочередно нажимать все кнопки.
3. Проверить верную работу кнопок.

Ожидаемый результат:

Корректная работа кнопок.

Фактический результат:

Корректная работа кнопок.

Чек-лист для полей ввода

|  |  |  |
| --- | --- | --- |
| Проверка | Ожидаемый результат | Фактический результат |
| Проверка отрисовки игрового поля | Должно открыться окно с отрисованным игровым полем и виджетами | Открылось окно с отрисованным игровым полем и виджетами |
| Проверка хода | Запрет выставления троек в первом этапе игры. | Запрет выставления троек в первом этапе игры. |
| Проверка окончания игры | Появление предупреждения об окончании игры, открытие окна с виджетами для выхода из игры или перезапуском игры. | Появление предупреждения об окончании игры, открытие окна с виджетами для выхода из игры или перезапуском игры. |
| Проверка правильности отрисовки всех виджетов и изображений. | Корректная работа всех изображений и виджетов. | Корректная работа всех изображений и виджетов. |
| Проверка хода компьютера. | Ход компьютера отображается правильно. Компьютер соблюдает правила игры. | Ход компьютера отображается правильно. Компьютер соблюдает правила игры. |
| Проверка корректности работы всех кнопок в соответствии с архитектурой. | Корректная работа кнопок. | Корректная работа кнопок. |

1.4.5 Выводы

На основе проведенных тестов сделаны следующие выводы:

– Программа успешно прошла все тесты и работает корректно.

**2. Источники, использованные при разработке**

1. Руководство по программированию на Tkinter и Python [Электронный ресурс]. – Режим доступа: для всех пользователей. – URL: <https://metanit.com/python/tkinter/?ysclid=lq9y8691w1850842283> (дата обращения: 10.07.24).

2. Документация Python 3.12.1. [Graphical User Interfaces with Tk](https://docs.python.org/3/library/tk.html) [Электронный ресурс]. – Режим доступа: для всех пользователей. – URL: https://docs.python.org/3/library/tk.html (дата обращения: 12.07.24).

3. 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: 20.07.24).

4. Материал из Википедии – свободной энциклопедии; оригинальные правила игры[Электронный ресурс]. – Режим доступа: для всех пользователей. – URL: <https://ru.wikipedia.org/wiki/Color_Lines> (дата обращения 24.07.24)

**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«Го-Бан»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

**Руководство программиста**

Р.02069337. 22/2359-27 РП-<01>

Листов: 22

**Исполнитель:**

студент гр. ИСТбд-32

Быкин Вадим Дмитриевич

« » 202\_\_ г.

202\_\_ г.

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Программа предназначена для игры в компьютерную игру «Го-Бан»

Краткие правила игры:

* Игра ведётся на поле 8х8 в два этапа. У игроков по 12 камней.
* В первом этапе игры игроки по очереди выставляют свои камни на поле (по одному за ход).
* В первом этапе игры существует возможность выставить пять своих камней в ряд по горизонтали, вертикали и диагонали.
* Во втором этапе, игроки передвигают свои камни.
* Атаковать или перепрыгивать через камни противника нельзя.
* Игра заканчивается, когда у какого-либо игрока собрана хотя бы один ряд из пяти камней.

Цель игры – собрать пять камней в ряд по горизонтали, вертикали или диагонали.

**1.2 Условия, необходимые для использования приложения**

Необходимы следующие библиотеки для использования приложения: tkinter, PIL, sqlite3, math.

Требования к операционной системе: Windows 10 и выше.

Требования к платформе: любая современная платформа с поддержкой Python версии 3.9

**2. Характеристики программы**

**2.1 Характеристики приложения**

Количество значимых строк кода: 739.

Количество алгоритмов: 28.

Используемые библиотеки:

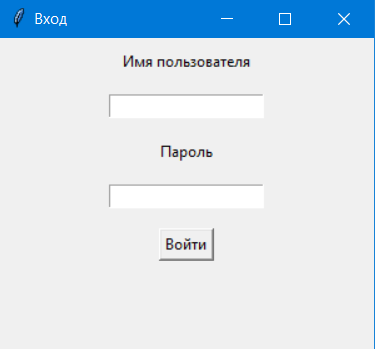
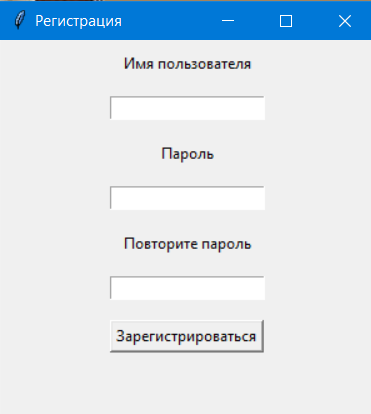
1. tkinter – библиотека для разработки графического интерфейса; в приложении используется для создания графического окна с виджетами;
2. PIL – библиотека языка Python (версии 2), предназначенная для работы с растровой графикой; в приложении библиотека используется для использования изображений библиотекой tkinter;
3. sqlite3 – это встроенная база данных, которая предоставляет мощные возможности хранения и управления данными внутри приложений.
4. math – эта библиотека в Python обеспечивает доступ к некоторым математическим функциям и константам, которые можно использовать в коде для различных математических вычислений.

Описание приложения:

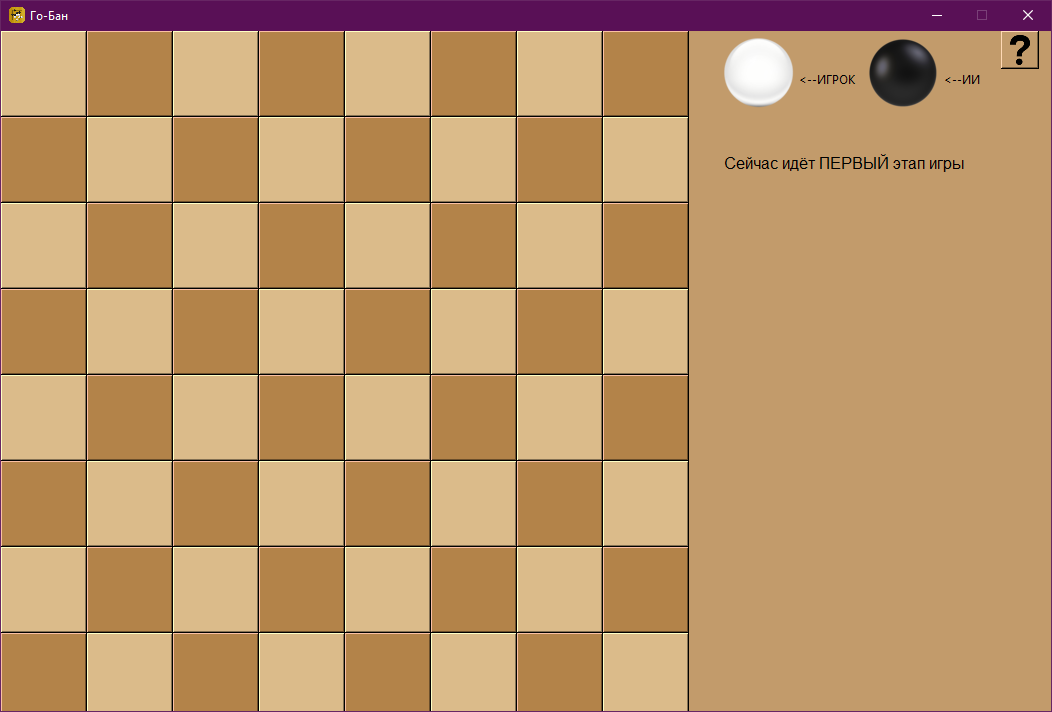
При запуске приложения появляется окно главного меню.

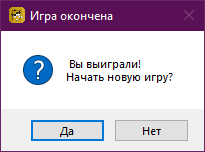
Пользователю предоставляется на выбор три действия: при нажатии на кнопку “Начать игру” откроется окно с выбором регистрации или авторизации, а также кнопка возвращения назад в главное меню; при нажатии на кнопку “Правила игры” пользователь сможет увидеть правила игры; при нажатии на кнопку “Выход” приложение завершит свою работу.

После авторизации/регистрации откроется окно с игрой, в котором отрисовано поле.



Справа от игрового поля пользователь видит всплывающие сообщения во время игры и кнопку. При завершении игры открывается новое окно, предлагающее начать новую игру или выйти.



**2.2 Особенности реализации приложения**

Использованные в программе структуры данных:

1. строки – главная структура данных в приложении, поскольку работа с файлами осуществляется на основе строкового представления данных; хранят содержимое файлов, их названия, кодировки и др;
2. списки – структура данных, применяемая для хранения путей к выбранным файлам, соответствующие им кодировки для корректной обработки;
3. целые числа – структура данных, необходимая для нумерации строк, записываемых в новый файл, а также нумерации выбранных файлов при их удалении в автоматическом режиме работы;
4. булевые значения – true/false – структура данных, используемая для проверки достижения конца файла, заполнения пользователем всех необходимых полей, наличия ошибок в файле.
5. двумерные массивы - структура данных, используемая для представления поля игры и работы над ним.

**3. Обращение к программе**

Функции:

Class Menu

1. \_\_init\_\_ - Инициализация;
2. start\_of\_the\_prog – Открытие и отрисовка главного меню;
3. changing\_lbl – Отрисовка привил игры;
4. new\_game – Отрисовка кнопок Вход, Регистрация, и тд;
5. back – функция для кнопки возвращения в главное меню;
6. start\_game\_without\_user – функция старта игры без регистрации;
7. overwrite\_user – функция обработчика файла;
8. CreateNewUser – функция проверки правильного заполнения в окне регистрации;
9. Register – функция отрисовки окна регистрации;
10. CheckExist - функция проверки правильного заполнения в окне входа;
11. Enter - функция отрисовки окна входа;

Class Gobang

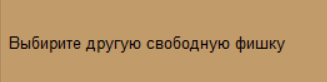
1. \_\_init\_\_ - инициализация;
2. rules – функция отрисовки окна с правилами во время игры;
3. colour\_change – функция перекраски игрового поля
4. change\_part – функция перехода во второй этап игры
5. click – функция обработки хода (клика) игрока
6. get\_possible\_move\_part\_1 – функция получения возможных ходов компьютера для первого этапа игры
7. get\_possible\_move\_part\_2 – функция получения возможных ходов компьютера для второго этапа игры
8. counter\_sequences – функция, считывающая последовательности из камней игрока/компьютера
9. search\_sequence\_player – функция рассчитывающая бонус за блокировку последовательностей из камней игрока для компьютера
10. evaluation\_function – расчётная функция
11. minimax – функция минимакса
12. computer\_move – функция обработки хода компьютера во втором этапе игры
13. find\_all\_lines – функция поиска линий игрока/компьютера
14. find\_all\_lines\_sequence – обработчик одной линии поля
15. is\_game\_over – функция проверки окончания игры

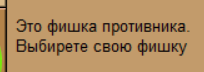
Используемые библиотеки:

1. tkinter – библиотека для разработки графического интерфейса; в приложении используется для создания графического окна с виджетами;
2. PIL – библиотека языка Python (версии 2), предназначенная для работы с растровой графикой; в приложении библиотека используется для использования изображений библиотекой tkinter;
3. s qlite3 - это встроенная база данных, которая предоставляет мощные возможности хранения и управления данными внутри приложений.
4. math – эта библиотека в Python обеспечивает доступ к некоторым математическим функциям и константам, которые можно использовать в коде для различных математических вычислений.

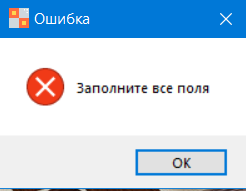
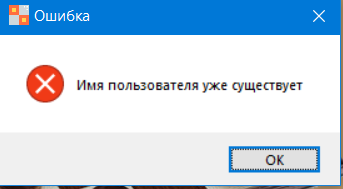
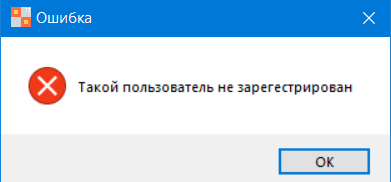
**4. Сообщения**

При попытке сходить не в соответствии с правилами игры, пользователь видит сообщение:





Если во время регистрации пользователь ошибается, от видит предупреждения:

**Текст программы:**

from tkinter import \*  
import tkinter.messagebox as mb  
from PIL import ImageTk  
import math  
  
class Menu(Tk):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.start\_of\_the\_prog()  
  
 def start\_of\_the\_prog(self): # отрисовка меню и всех кнопок  
 self.wm\_iconphoto(False, ImageTk.PhotoImage(file="Gobang/go-ban\_icon.png", master=self))  
 self.title('Го-Бан')  
 self.configure(bg="#c29b6b")  
 w = 800  
 h = 450  
 ws = self.winfo\_screenwidth()  
 hs = self.winfo\_screenheight()  
 x = (ws / 2) - (w / 2)  
 y = (hs / 2) - (h / 2)  
 self.geometry('%dx%d+%d+%d' % (w, h, x, y))  
 self.resizable(False, False)  
 self.change\_lbl = False  
 self.image = ImageTk.PhotoImage(file="Gobang/go-ban.png", master=self)  
 self.lbl\_front = Label(self, image=self.image, bg="#c29b6b", font='Areal 14', justify="left",  
 wraplength=525)  
 self.lbl\_front.place(x=0, y=-15)  
 lbl\_name = Label(self, bg="#c29b6b", text="Игра Го-Бан", font='Arial 24')  
 lbl\_name.place(x=545, y=40)  
 self.btn\_startgame = Button(self, bg="#c29b6b", text="Начать игру", font="Arial 14", command=self.new\_game)  
 self.btn\_startgame.place(x=583, y=240)  
 self.btn\_rule = Button(self, bg="#c29b6b", text="Правила игры", font="Arial 14", command=self.changing\_lbl)  
 self.btn\_rule.place(x=575, y=285)  
 self.btn\_out = Button(self, bg="#c29b6b", text="Выход", font="Arial 14", command=lambda: exit(0))  
 self.btn\_out.place(x=602, y=330)  
  
 def changing\_lbl(self): # смена правил игры на иконку го и наоборот  
 if self.change\_lbl:  
 self.lbl\_front.configure(image=self.image, text="")  
 self.change\_lbl = False  
 return  
 if not self.change\_lbl:  
 with open('Gobang/rules.txt', 'r', encoding='utf8') as file:  
 rules = file.read()  
 file.close()  
 self.lbl\_front.configure(image="", text=rules)  
 self.change\_lbl = True  
 return  
  
 def new\_game(self): # кнопка НАЧАТЬ ИГРУ  
 self.btn\_guest = Button(self, bg="#c29b6b", text="Продолжить как гость", font="Arial 14", command=  
 self.start\_game\_without\_user)  
 self.btn\_enter = Button(self, bg="#c29b6b", text="Войти в профиль", font="Arial 14", command=self.enter)  
 self.btn\_regist = Button(self, bg="#c29b6b", text="Зарегистрироваться", font="Arial 14",  
 command=self.register)  
 self.btn\_back = Button(self, bg="#c29b6b", text="Назад", font="Arial 14", command=self.back)  
 self.btn\_guest.place(x=542, y=240)  
 self.btn\_enter.place(x=565, y=285)  
 self.btn\_regist.place(x=550, y=330)  
 self.btn\_back.place(x=610, y=375)  
  
 def back(self): # кнопка НАЗАД  
 self.btn\_guest.destroy()  
 self.btn\_enter.destroy()  
 self.btn\_regist.destroy()  
 self.btn\_back.destroy()  
  
 def start\_game\_without\_user(self): # кнопка ПРОДОЛЖИТЬ КАК ГОСТЬ  
 self.withdraw()  
 with open("Gobang/User.txt", 'r+') as f:  
 f.truncate()  
 Gobang(self)  
 self.deiconify()  
  
 def overwrite\_user(self, username):  
 with open("Gobang/User.txt", "w+") as f2:  
 f2.truncate()  
 f2.write(username)  
  
 def CreateNewUser(self, username, password, password\_again): # проверка на правильное заполнение полей в окне регистрации  
 if username == '' or password == '' or password\_again == '':  
 msg = 'Заполните все поля'  
 mb.showerror("Ошибка", msg)  
 else:  
 try:  
 f1 = open('Gobang/Users.txt', 'r+')  
 text = f1.read().split()  
 for i in text:  
 if username == i.split(':')[0]:  
 msg = 'Имя пользователя уже существует'  
 mb.showerror("Ошибка", msg)  
 return  
 else:  
 if password != password\_again:  
 msg = 'Пароли не совпадают'  
 mb.showerror("Ошибка", msg)  
 return  
 else:  
 f1.write(username + ':' + password + ':' + '0' + '\n')  
 self.overwrite\_user(username)  
 f1.close()  
 self.window\_Reg.destroy()  
 self.withdraw()  
 Gobang(self)  
 self.deiconify()  
 return  
 except:  
 f1 = open('Gobang/Users.txt', 'w')  
 if password != password\_again:  
 msg = 'Пароли не совпадают'  
 mb.showerror("Ошибка", msg)  
 return  
 else:  
 f1.write(username + ':' + password + ':' + '0' + '\n')  
 self.overwrite\_user(username)  
 f1.close()  
 self.window\_Reg.destroy()  
 self.back()  
 self.withdraw()  
 Gobang(self)  
 self.deiconify()  
 return  
  
 def register(self): # отрисовка окна регистрации  
 self.window\_Reg = Tk()  
 self.window\_Reg.title('Регистрация')  
 self.window\_Reg.geometry('300x300')  
 self.window\_Reg.eval('tk::PlaceWindow . center')  
 username\_label = Label(self.window\_Reg, text='Имя пользователя', )  
 username\_entry = Entry(self.window\_Reg)  
 password\_label = Label(self.window\_Reg, text='Пароль')  
 password\_entry = Entry(self.window\_Reg)  
 password\_label\_confirm = Label(self.window\_Reg, text='Повторите пароль')  
 password\_entry\_confirm = Entry(self.window\_Reg)  
 send\_btn = Button(self.window\_Reg, text='Зарегистрироваться', command=lambda:  
 self.CreateNewUser(username\_entry.get(), password\_entry.get(), password\_entry\_confirm.get(), ))  
 username\_label.pack(padx=10, pady=8)  
 username\_entry.pack(padx=10, pady=8)  
 password\_label.pack(padx=10, pady=8)  
 password\_entry.pack(padx=10, pady=8)  
 password\_label\_confirm.pack(padx=10, pady=8)  
 password\_entry\_confirm.pack(padx=10, pady=8)  
 send\_btn.pack(padx=10, pady=8)  
 self.window\_Reg.mainloop()  
  
 def CheckExist(self, username, password): # проверка на правильность заполнения полей в окне входа  
 if username == '' or password == '':  
 msg = 'Заполните все поля'  
 mb.showerror("Ошибка", msg)  
 else:  
 f1 = open('Gobang/Users.txt', 'r')  
 text = f1.read().split()  
 for i in text:  
 if i.split(':')[0] == username:  
 if password == i.split(':')[1]:  
 self.overwrite\_user(username)  
 f1.close()  
 self.window\_enter.destroy()  
 self.withdraw()  
 Gobang(self)  
 self.deiconify()  
 return  
 else:  
 msg = 'Пароль не совпадает'  
 mb.showerror("Ошибка", msg)  
 return  
 msg = 'Такой пользователь не зарегестрирован'  
 mb.showerror("Ошибка", msg)  
 self.window\_enter.destroy()  
 self.register()  
 return  
  
 def enter(self): # функция отрисовки окна входа  
 self.window\_enter = Tk()  
 self.window\_enter.title('Вход')  
 self.window\_enter.geometry('300x250')  
 self.window\_enter.eval('tk::PlaceWindow . center')  
 username\_label = Label(self.window\_enter, text='Имя пользователя', )  
 username\_entry = Entry(self.window\_enter)  
 password\_label = Label(self.window\_enter, text='Пароль')  
 password\_entry = Entry(self.window\_enter)  
 send\_btn = Button(self.window\_enter, text='Войти', command=lambda:  
 self.CheckExist(username\_entry.get(), password\_entry.get()))  
  
 username\_label.pack(padx=10, pady=8)  
 username\_entry.pack(padx=10, pady=8)  
 password\_label.pack(padx=10, pady=8)  
 password\_entry.pack(padx=10, pady=8)  
 send\_btn.pack(padx=10, pady=8)  
  
 self.window\_enter.mainloop()  
  
  
class Gobang(Toplevel):  
 def \_\_init\_\_(self, parent):  
 super().\_\_init\_\_(parent)  
 self.reload() # запуски игры  
  
 def reload(self): # запуск/перезагрузка игры  
 self.wm\_iconphoto(False, ImageTk.PhotoImage(file="Gobang/go-ban\_icon.png"))  
 self.title('Го-Бан')  
 self.configure(bg="#c29b6b")  
 w = 1050  
 h = 680  
 ws = self.winfo\_screenwidth()  
 hs = self.winfo\_screenheight()  
 x = (ws/2) - (w/2)  
 y = (hs/2) - (h/2)  
 self.geometry('%dx%d+%d+%d' % (w, h, x, y))  
 self.resizable(False, False)  
 self.flag\_is\_first\_part\_game = True # флаг этапа игры (первый этап - расставление, второй - передвижение)  
 self.sequence\_weights = {2: 200, 3: 1000, 4: 5000, 5: 100000}  
 self.choice = False # флаг, выбрана ли фишка во втором этапе  
 self.count\_figur = 0 # счетчик поставленных фишек для перехода во второй этап игры  
 self.field = [["-", "-", "-", "-", "-", "-", "-", "-", "-", "-"],  
 ["|", 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, 0, 0, 0, 0, 0, "|"],  
 ["|", 0, 0, 0, 0, 0, 0, 0, 0, "|"], # поле игры ("-" и "|" - ограничение поля, 0 - пустое поле, 1 - игрок, 2 - компьютер)  
 ["|", 0, 0, 0, 0, 0, 0, 0, 0, "|"],  
 ["|", 0, 0, 0, 0, 0, 0, 0, 0, "|"],  
 ["|", 0, 0, 0, 0, 0, 0, 0, 0, "|"],  
 ["-", "-", "-", "-", "-", "-", "-", "-", "-", "-"]]  
 self.weight\_matrix = [[1, 2, 3, 4, 4, 3, 2, 1],  
 [2, 3, 4, 5, 5, 4, 3, 2],  
 [3, 4, 5, 6, 6, 5, 4, 3],  
 [4, 5, 6, 7, 7, 6, 5, 4],  
 [4, 5, 6, 7, 7, 6, 5, 4],  
 [3, 4, 5, 6, 6, 5, 4, 3],  
 [2, 3, 4, 5, 5, 4, 3, 2],  
 [1, 2, 3, 4, 4, 3, 2, 1]]  
 self.image1 = ImageTk.PhotoImage(file="Gobang/black.png")  
 self.image2 = ImageTk.PhotoImage(file="Gobang/white.png")  
 self.image3 = ImageTk.PhotoImage(file="Gobang/empty.png")  
 image\_settings = ImageTk.PhotoImage(file="Gobang/question\_mark.png")  
 self.button\_help = Button(self, bg="#c29b6b", image=image\_settings, command=self.rules)  
 self.button\_help.place(x=1000, y=0)  
 lbl\_2 = Label(self, bg="#c29b6b", text='<--ИГРОК')  
 lbl\_1 = Label(self, bg="#c29b6b", text='<--ИИ')  
 lbl\_2.place(x=795, y=38)  
 lbl\_1.place(x=940, y=38)  
 panel1 = Label(self, bg="#c29b6b", image=self.image1)  
 panel2 = Label(self, bg="#c29b6b", image=self.image2)  
 panel1.place(x=860, y=0)  
 panel2.place(x=715, y=0)  
 self.lbl\_part = Label(self, bg="#c29b6b", text="Сейчас идёт ПЕРВЫЙ этап игры", font='Areal 12')  
 self.lbl\_part.place(x=720, y=120)  
 self.lbl\_popup = Label(self, bg="#c29b6b", text='', font='Areal 12', justify='left', wraplength=278)  
 self.lbl\_popup.place(x=720, y=175)  
 self.b = [] # заготовка поля из матрицы кнопок для визуала  
 for i in range(10):  
 self.b.append(i)  
 self.b[i] = []  
 for j in range(10):  
 self.b[i].append(j)  
 if (0 < i < 9) and (0 < j < 9): # выводим только поле размером 8х8 с сохранением координат 9х9  
 self.b[i][j] = Button(self, height=80, width=80, image=self.image3,  
 command=lambda row=i, col=j: self.click(row, col))  
 self.b[i][j].grid(row=i, column=j)  
 self.colour\_change()  
 self.wait\_window(self)  
  
 def rules(self): # окно с правилами во время игры  
 self.button\_help.configure(command="")  
 rule = Toplevel(self)  
 rule.wm\_iconphoto(False, ImageTk.PhotoImage(file="Gobang/icon.png"))  
 rule.title("Правила игры")  
 rule.geometry("462x330")  
 rule.configure(bg="#c29b6b")  
 rule.resizable(False, False)  
 with open('Gobang/rules.txt', 'r', encoding='utf8') as file:  
 rules = file.read()  
 lbl = Label(rule, bg="#c29b6b", text=rules, font='Areal 12', justify="left", wraplength=460)  
 lbl.place(x=0, y=0)  
  
 def quit\_window():  
 self.button\_help.config(command=self.rules)  
 rule.destroy()  
 rule.protocol("WM\_DELETE\_WINDOW", quit\_window)  
  
 def colour\_change(self): # перекраска поля  
 for i in range(1, 9):  
 for j in range(1, 9):  
 if (i + j) % 2 == 0:  
 self.b[i][j].configure(bg="#dbbb8a")  
 else:  
 self.b[i][j].configure(bg="#b38349")  
 return  
  
 def change\_part(self): # смена игры из первой части во второую  
 if self.count\_figur == 12:  
 self.count\_figur = 0  
 self.flag\_is\_first\_part\_game = False  
 self.lbl\_part.configure(text='Сейчас идёт ВТОРОЙ этап игры')  
 return  
  
 def click(self, row, col): # обработчик кнопок поля  
 try:  
 self.lbl\_popup.configure(text='')  
 if self.flag\_is\_first\_part\_game: # если идёт первый этап игры  
 self.colour\_change()  
 count = 0  
 if self.field[row][col] == 0:  
 self.b[row][col].configure(image=self.image2)  
 self.field[row][col] = 1  
 count += 1  
 if self.is\_game\_over(True):  
 win = mb.askquestion('Игра окончена', f' Вы выиграли!\n Начать новую игру?')  
 self.reload() if win == 'yes' else self.destroy()  
 if count == 1:  
 count = 0  
 self.computer\_move(2, 1)  
 if self.is\_game\_over(False):  
 win = mb.askquestion('Игра окончена', f' Вы проиграли!\n Начать новую игру?')  
 self.reload() if win == 'yes' else self.destroy()  
 self.count\_figur += 1  
 self.change\_part()  
 else: # если второй этап игры  
 if not self.choice: # выбор своей фишки для хода ею  
 if self.field[row][col + 1] != 0 and self.field[row][col - 1] != 0 \  
 and self.field[row + 1][col] != 0 and self.field[row - 1][col] != 0 \  
 and self.field[row][col] == 1:  
 self.lbl\_popup.configure(text='Выбирите другую свободную фишку')  
 elif self.field[row][col] == 2:  
 self.lbl\_popup.configure(text='Это фишка противника.\nВыбирете свою фишку')  
 elif self.field[row][col] == 1:  
 self.colour\_change()  
 self.row2, self.col2 = row, col  
 self.b[row][col].configure(bg='yellow')  
 self.choice = True  
 elif self.row2 == row and self.col2 == col: # отменить выбранную фишку  
 self.choice = False  
 self.colour\_change()  
 else: # ход выбранной фишкой  
 if (abs(row - self.row2) + abs(col - self.col2) == 1) and self.field[row][col] == 0:  
 self.b[row][col].configure(image=self.image2)  
 self.field[row][col] = 1  
 self.b[self.row2][self.col2].configure(image=self.image3)  
 self.colour\_change()  
 self.field[self.row2][self.col2] = 0  
 self.choice = False  
 if self.is\_game\_over(True):  
 win = mb.askquestion('Игра окончена', f' Вы выиграли!\n Начать новую игру?')  
 self.reload() if win == 'yes' else self.destroy()  
 self.computer\_move(2, 1)  
 if self.is\_game\_over(False):  
 win = mb.askquestion('Игра окончена', f' Вы проиграли!\n Начать новую игру?')  
 self.reload() if win == 'yes' else self.destroy()  
 return  
 except Exception as e:  
 pass  
  
 def get\_possible\_move\_part\_1(self, maximizing\_player, computer\_symbol, player\_symbol): # возможные ходы компа часть 1  
 lines = self.find\_all\_lines(maximizing\_player)  
 moves = []  
 if maximizing\_player:  
 symbol = player\_symbol  
 else:  
 symbol = computer\_symbol  
 for item in lines:  
 row1, col1 = item[0]  
 row2, col2 = item[1]  
 lengh = len(item)  
 if self.field[row1][col1] == symbol:  
 if col2 - col1 == 1 and row1 == row2:  
 if self.field[row1][col1 - 1] == 0 and not((row1, col1 - 1) in moves):  
 moves.append((row1, col1 - 1))  
 if self.field[row1][col1 + lengh] == 0 and not((row1, col1 + lengh) in moves):  
 moves.append((row1, col1 + lengh))  
 elif row2 - row1 == 1 and col1 == col2:  
 if self.field[row1 - 1][col1] == 0 and not((row1 - 1, col1) in moves):  
 moves.append((row1 - 1, col1))  
 if self.field[row1 + lengh][col1] == 0 and not((row1 + lengh, col1) in moves):  
 moves.append((row1 + lengh, col1))  
 elif row2 - row1 == 1 and col2 - col1 == 1:  
 if self.field[row1 - 1][col1 - 1] == 0 and not((row1 - 1, col1 - 1) in moves):  
 moves.append((row1 - 1, col1 - 1))  
 if self.field[row1 + lengh][col1 + lengh] == 0 and not((row1 + lengh, col1 + lengh) in moves):  
 moves.append((row1 + lengh, col1 + lengh))  
 else:  
 if self.field[row1 - 1][col1 + 1] == 0 and not((row1 - 1, col1 + 1) in moves):  
 moves.append((row1 - 1, col1 + 1))  
 if self.field[row1 + lengh][col1 - lengh] == 0 and not((row1 + lengh, col1 - lengh) in moves):  
 moves.append((row1 + lengh, col1 - lengh))  
 if not moves:  
 for row in range(1, 9):  
 for col in range(1, 9):  
 if self.field[row - 1][col] == self.field[row + 1][col] == symbol and self.field[row][col] == 0:  
 moves.append((row, col))  
 if self.field[row][col - 1] == self.field[row][col + 1] == symbol and self.field[row][col] == 0:  
 moves.append((row, col))  
 if self.field[row - 1][col - 1] == self.field[row + 1][col + 1] == symbol and self.field[row][col] == 0:  
 moves.append((row, col))  
 if self.field[row - 1][col + 1] == self.field[row + 1][col - 1] == symbol and self.field[row][col] == 0:  
 moves.append((row, col))  
 if not moves:  
 if row < 5 and col < 5 and self.field[row][col] == symbol:  
 if self.field[5][col] == 0 and not((5, col) in moves):  
 moves.append((5, col))  
 if self.field[row][5] == 0 and not((row, 5) in moves):  
 moves.append((row, 5))  
 if self.field[5][5] == 0 and row == col == 4:  
 moves.append((5, 5))  
 elif self.field[row + 2][col + 2] == 0 and not((row + 2, col + 2) in moves):  
 moves.append((row + 2, col + 2))  
 if row < 5 <= col and self.field[row][col] == symbol:  
 if self.field[5][col] == 0 and not((5, col) in moves):  
 moves.append((5, col))  
 if self.field[row][4] == 0 and not((row, 4) in moves):  
 moves.append((row, 4))  
 if self.field[5][4] == 0 and row == 4 and col == 5:  
 moves.append((5, 4))  
 elif self.field[row + 2][col - 2] == 0 and not((row + 2, col - 2) in moves):  
 moves.append((row + 2, col - 2))  
 if row >= 5 > col and self.field[row][col] == symbol:  
 if self.field[4][col] == 0 and not((4, col) in moves):  
 moves.append((4, col))  
 if self.field[row][5] == 0 and not((row, 5) in moves):  
 moves.append((row, 5))  
 if self.field[4][5] == 0 and row == 5 and col == 4:  
 moves.append((4, 5))  
 elif self.field[row - 2][col + 2] == 0 and not((row - 2, col + 2) in moves):  
 moves.append((row - 2, col + 2))  
 if row >= 5 and col >= 5 and self.field[row][col] == symbol:  
 if self.field[4][col] == 0 and not((4, col) in moves):  
 moves.append((4, col))  
 if self.field[row][4] == 0 and not((row, 4) in moves):  
 moves.append((row, 4))  
 if self.field[4][4] == 0 and row == col == 5:  
 moves.append((4, 4))  
 elif self.field[row - 2][col - 2] == 0 and not((row - 2, col - 2) in moves):  
 moves.append((row - 2, col - 2))  
 return moves  
  
 def get\_possible\_move\_part\_2(self, maximizing\_player, computer\_symbol, player\_symbol): # возможные ходы компа часть 2  
 possible\_move = []  
 for row in range(1, 9):  
 for col in range(1, 9):  
 if maximizing\_player:  
 if self.field[row + 1][col] == 0 and self.field[row][col] == computer\_symbol:  
 possible\_move.append([(row, col), (row + 1, col)])  
 if self.field[row][col + 1] == 0 and self.field[row][col] == computer\_symbol:  
 possible\_move.append([(row, col), (row, col + 1)])  
 if self.field[row - 1][col] == 0 and self.field[row][col] == computer\_symbol:  
 possible\_move.append([(row, col), (row - 1, col)])  
 if self.field[row][col - 1] == 0 and self.field[row][col] == computer\_symbol:  
 possible\_move.append([(row, col), (row, col - 1)])  
 else:  
 if self.field[row + 1][col] == 0 and self.field[row][col] == player\_symbol:  
 possible\_move.append([(row, col), (row + 1, col)])  
 if self.field[row][col + 1] == 0 and self.field[row][col] == player\_symbol:  
 possible\_move.append([(row, col), (row, col + 1)])  
 if self.field[row - 1][col] == 0 and self.field[row][col] == player\_symbol:  
 possible\_move.append([(row, col), (row - 1, col)])  
 if self.field[row][col - 1] == 0 and self.field[row][col] == player\_symbol:  
 possible\_move.append([(row, col), (row, col - 1)])  
 return possible\_move  
  
 def counter\_sequences(self, maximizing\_player): # счетчик последовательностей игрока/компьютера  
 sequences = {2: 0, 3: 0, 4: 0}  
 lines = self.find\_all\_lines(maximizing\_player)  
 for item in lines:  
 if len(item) == 2:  
 sequences[2] += 1  
 if len(item) == 3:  
 sequences[3] += 1  
 if len(item) == 4:  
 sequences[4] += 1  
 return sequences  
  
 def search\_sequence\_player(self, computer\_symbol, player\_symbol): # бонус за блокировку последовательностей игрока  
 score = 0  
 lines = self.find\_all\_lines(True)  
 for item in lines:  
 row1, col1 = item[0]  
 row2, col2 = item[1]  
 lengh = len(item)  
 if col2 - col1 == 1 and row1 == row2:  
 if self.field[row1][col1 - 1] == computer\_symbol:  
 if self.field[row1][col1 + lengh] == computer\_symbol:  
 score += 3750 if lengh >= 4 else 1250 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1][col1 - 1] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 if self.field[row1][col1 + lengh] == computer\_symbol:  
 if self.field[row1][col1 - 1] == computer\_symbol:  
 score += 3750 if lengh >= 4 else 1250 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1][col1 + lengh] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif row2 - row1 == 1 and col1 == col2:  
 if self.field[row1 - 1][col1] == computer\_symbol:  
 if self.field[row1 + lengh][col1] == computer\_symbol:  
 score += 3750 if lengh >= 4 else 1250 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1 - 1][col1] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 if self.field[row1 + lengh][col1] == computer\_symbol:  
 if self.field[row1 - 1][col1] == computer\_symbol:  
 score += 3750 if lengh >= 4 else 1250 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1 + lengh][col1] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif row2 - row1 == 1 and col2 - col1 == 1:  
 if self.field[row1 - 1][col1 - 1] == computer\_symbol:  
 if self.field[row1 + lengh][col1 + lengh] == computer\_symbol:  
 score += 3750 if lengh >= 4 else 1250 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1 - 1][col1 - 1] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 if self.field[row1 + lengh][col1 + lengh] == computer\_symbol:  
 if self.field[row1 + lengh][col1 + lengh] == computer\_symbol:  
 score += 3750 if lengh >= 4 else 1250 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1 + lengh][col1 + lengh] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 else:  
 if self.field[row1 - 1][col1 + 1] == computer\_symbol:  
 if self.field[row1 + lengh][col1 - lengh] == computer\_symbol:  
 score += 3700 if lengh >= 4 else 1200 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1 - 1][col1 + 1] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 if self.field[row1 + lengh][col1 - lengh] == computer\_symbol:  
 if self.field[row1 - 1][col1 + 1] == computer\_symbol:  
 score += 3750 if lengh >= 4 else 1250 if lengh == 3 else 750  
 else:  
 score += 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 elif self.field[row1 + lengh][col1 - lengh] == player\_symbol:  
 score -= 7000 if lengh >= 4 else 3500 if lengh == 3 else 1000  
 return score  
  
 def evaluation\_function(self, computer\_symbol, player\_symbol): # расчётная функция  
 score = 0  
 player\_sequences = self.counter\_sequences(True)  
 computer\_sequences = self.counter\_sequences(False)  
 for length, count in computer\_sequences.items(): # 1 оценка: посчет последовательностей фишек  
 score += self.sequence\_weights.get(length, 0) \* count  
 for length, count in player\_sequences.items():  
 score -= self.sequence\_weights.get(length, 0) \* count  
 for row in range(len(self.field)): # 2 оценка: бонус за позиции на основе весовой матрицы  
 for col in range(len(self.field)):  
 if self.field[row][col] == computer\_symbol:  
 score += self.weight\_matrix[row - 1][col - 1]  
 elif self.field[row][col] == player\_symbol:  
 score -= self.weight\_matrix[row - 1][col - 1]  
 score += self.search\_sequence\_player(computer\_symbol, player\_symbol) # 3 оценка: бонус за блокировку игрока  
 return score  
  
 def minimax(self, depth, alpha, beta, maximizing\_player, computer\_symbol, player\_symbol): # минимакс  
 if depth == 0 or self.is\_game\_over(maximizing\_player):  
 return self.evaluation\_function(computer\_symbol, player\_symbol), None  
 if self.flag\_is\_first\_part\_game:  
 possible\_move = self.get\_possible\_move\_part\_1(maximizing\_player, computer\_symbol, player\_symbol)  
 else:  
 possible\_move = self.get\_possible\_move\_part\_2(maximizing\_player, computer\_symbol, player\_symbol)  
 if maximizing\_player:  
 max\_eval = -math.inf  
 best\_move = None  
 if self.flag\_is\_first\_part\_game:  
 for move in possible\_move:  
 row, col = move  
 self.field[row][col] = computer\_symbol  
 eval, \_ = self.minimax(depth - 1, alpha, beta, False, computer\_symbol, player\_symbol)  
 self.field[row][col] = 0  
 if eval > max\_eval:  
 max\_eval = eval  
 best\_move = move  
 alpha = max(alpha, eval)  
 if beta <= alpha:  
 break  
 return max\_eval, best\_move  
 else:  
 for move1, move2 in possible\_move:  
 row1, col1 = move1  
 row2, col2 = move2  
 self.field[row1][col1] = 0  
 self.field[row2][col2] = computer\_symbol  
 eval, \_ = self.minimax(depth - 1, alpha, beta, False, computer\_symbol, player\_symbol)  
 self.field[row1][col1] = computer\_symbol  
 self.field[row2][col2] = 0  
 if eval > max\_eval:  
 max\_eval = eval  
 best\_move = (move1, move2)  
 alpha = max(alpha, eval)  
 if beta <= alpha:  
 break  
 return max\_eval, best\_move  
 else:  
 min\_eval = math.inf  
 best\_move = None  
 if self.flag\_is\_first\_part\_game:  
 for move in possible\_move:  
 row, col = move  
 self.field[row][col] = player\_symbol  
 eval, \_ = self.minimax(depth - 1, alpha, beta, True, computer\_symbol, player\_symbol)  
 self.field[row][col] = 0  
 if eval < min\_eval:  
 min\_eval = eval  
 best\_move = move  
 beta = min(beta, eval)  
 if beta <= alpha:  
 break  
 return min\_eval, best\_move  
 else:  
 for move1, move2 in possible\_move:  
 row1, col1 = move1  
 row2, col2 = move2  
 self.field[row1][col1] = 0  
 self.field[row2][col2] = player\_symbol  
 eval, \_ = self.minimax(depth - 1, alpha, beta, True, computer\_symbol, player\_symbol)  
 self.field[row1][col1] = player\_symbol  
 self.field[row2][col2] = 0  
 if eval > min\_eval:  
 min\_eval = eval  
 best\_move = (move1, move2)  
 alpha = max(alpha, eval)  
 if beta <= alpha:  
 break  
 return min\_eval, best\_move  
  
 def computer\_move(self, computer\_symbol, player\_symbol, depth=3): # ход компьютера  
 if self.flag\_is\_first\_part\_game:  
 \_, move = self.minimax(depth, -math.inf, math.inf, True, computer\_symbol, player\_symbol)  
 if move:  
 row, col = move  
 self.field[row][col] = computer\_symbol  
 self.b[row][col].configure(image=self.image1)  
 else:  
 \_, move = self.minimax(depth, -math.inf, math.inf, True, computer\_symbol, player\_symbol)  
 if move:  
 row1, col1 = move[0]  
 row2, col2 = move[1]  
 self.field[row1][col1] = 0  
 self.b[row1][col1].configure(image=self.image3)  
 self.field[row2][col2] = computer\_symbol  
 self.b[row2][col2].configure(image=self.image1)  
 return  
  
 def find\_all\_lines(self, maximizing\_player): # функция поиска линий игрока/компьютера  
 n = len(self.field)  
 lines = []  
 if maximizing\_player:  
 symbol = 1  
 else:  
 symbol = 2  
  
 def find\_lines\_in\_sequence(sequence, coords): # обработчик одной линии поля  
 current\_value = None  
 current\_length = 0  
 start\_index = 0  
 for i, value in enumerate(sequence):  
 if value == current\_value and value == symbol:  
 current\_length += 1  
 else:  
 if current\_length >= 3:  
 line\_coords = coords[start\_index:start\_index + current\_length]  
 lines.append(line\_coords)  
 current\_value = value  
 current\_length = 1 if value == symbol else 0  
 start\_index = i  
 if current\_length >= 3: # Проверка последней последовательности  
 line\_coords = coords[start\_index:start\_index + current\_length]  
 lines.append(line\_coords)  
  
 for start\_row in range(n): # Поиск линий на основных диагоналях (слева направо, сверху вниз)  
 diagonal = [] # налево вниз  
 diag\_coords = []  
 row, col = start\_row, 0  
 while row < n and col < n:  
 diagonal.append(self.field[row][col])  
 diag\_coords.append((row, col))  
 row += 1  
 col += 1  
 find\_lines\_in\_sequence(diagonal, diag\_coords)  
 for start\_col in range(1, n): # направо вверх  
 diagonal = []  
 diag\_coords = []  
 row, col = 0, start\_col  
 while row < n and col < n:  
 diagonal.append(self.field[row][col])  
 diag\_coords.append((row, col))  
 row += 1  
 col += 1  
 find\_lines\_in\_sequence(diagonal, diag\_coords)  
 for start\_row in range(n): # Поиск линий на побочных диагоналях (справа налево, сверху вниз)  
 diagonal = [] # напрво вниз  
 diag\_coords = []  
 row, col = start\_row, n - 1  
 while row < n and col >= 0:  
 diagonal.append(self.field[row][col])  
 diag\_coords.append((row, col))  
 row += 1  
 col -= 1  
 find\_lines\_in\_sequence(diagonal, diag\_coords)  
 for start\_col in range(n - 2, -1, -1): # налево вверх  
 diagonal = []  
 diag\_coords = []  
 row, col = 0, start\_col  
 while row < n and col >= 0:  
 diagonal.append(self.field[row][col])  
 diag\_coords.append((row, col))  
 row += 1  
 col -= 1  
 find\_lines\_in\_sequence(diagonal, diag\_coords)  
 for row in range(n): # Поиск горизонтальных линий (по строкам)  
 horizontal = []  
 horiz\_coords = []  
 for col in range(n):  
 horizontal.append(self.field[row][col])  
 horiz\_coords.append((row, col))  
 find\_lines\_in\_sequence(horizontal, horiz\_coords)  
 for col in range(n): # Поиск вертикальных линий (по столбцам)  
 vertical = []  
 vert\_coords = []  
 for row in range(n):  
 vertical.append(self.field[row][col])  
 vert\_coords.append((row, col))  
 find\_lines\_in\_sequence(vertical, vert\_coords)  
 return lines  
  
 def is\_game\_over(self, maximizing\_player): # функция проверки окончания игры  
 lines = self.find\_all\_lines(maximizing\_player)  
 for lengh in lines:  
 if len(lengh) >= 5:  
 return True  
 return False  
  
  
menu = Menu()  
menu.mainloop()