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:** «Болотуду»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337. 22/2359-27 ТЗ-01

Листов: 43

Руководитель разработки:

к. т. н., доцент

Шишкин Вадим Викторинович

« » 202\_\_ г.

Исполнитель:

студентка гр. ИСТбд-22

Афонин Константин

Сергеевич

« » 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 «Color Lines». 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: «Болотуду»

Р.02069337. 22/2359-27 ТЗ-01

Листов: 7

**Исполнитель**:

студентка гр. ИСТбд-22

Афонин Константин Сергеевич

« » 202\_\_ г.

202\_\_ г.

**Введение**

Курсовая работа представляет собой однопользовательское десктопное приложение, компьютерную игру «Болотуду»

Краткие правила игры:

* Игра ведётся на поле 5х6 в два этапа. У игроков по 12 фишек.
* В первом этапе игры игроки выставляю на поле по 2 фишки.
* В первом этапе игры запрещается выставление троек из своих фишек по горизонтали и вертикали.
* Во втором этапе, игроки передвигают свои фишки.
* Собрав троку из своих фишек, игрок может съесть фишку противника, находящуюся на продолжении построенной тройки.
* Игра заканчивается, когда у какого-либо игрока осталось 2 фишки.

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 «Болотуду» с графическим интерфейсом в среде Windows.

**2.2. Требования к функциональным характеристикам**

Приложение должно соответствовать следующим правилам игры:

Игра ведется на доске 5х6. Каждому игроку выдается набор из 12 камней своего цвета. Игра состоит из двух стадий.

На первой стадии игроки выставляют шашки, причем за один ход игрок выставляет сразу две своих шашки. На первой стадии запрещены ходы, в результате которых, появляется непрерывный ряд из трех камней по вертикали или горизонтали. Игру второй стадии начинает тот игрок, которому переходит очередь хода после завершения первой стадии.

Цель второй стадии - создание троек камней. Если игроку удалось построить такую тройку и к какому-либо её концу примыкает камень противника, то игрок, создавший тройку, снимает один камень, но только один, даже если к созданной тройке примыкает два камня, с двух концов. Снимать камень можно только в момент создания тройки, и только тот камень, который примыкает к только что созданной тройке. Тройку можно разрушить и вновь создать. Одной и той же тройкой можно пользовать многократно.

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 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. 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**

Указываются требования к работоспособности и способам восстановления при сбоях.

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

Операционная система: Windows 10

Используемые библиотеки: tkinter, PIL, sqlite3, random

Язык: Python 3.9

Среда разработки: PyCharm Community Edition 2023.1.2

**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

НЕ ПРАВИЛЬНО:

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 НЕ ПРАВИЛЬНО:

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 2023 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: «**Болотуду»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

**Пояснительная записка**

Р.02069337. 22/2359-27 ПЗ-01

Листов: 12

**Исполнитель**:

студентка гр. ИСТбд-22

Афонин Константин Сергеевич

« » 202\_\_ г.

202\_\_ г.

**Введение**

Программа предоставляет собой игру «Болотуду».

**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 7х8 для компьютера и 5х6 визуальное поле для игрока. А также модель поиска выгодной компьютеру клетки для преимущества над игроком.

Модель поля для компьютера:

["ы", "ы", "ы", "ы", "ы", "ы", "ы", "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", "ы", "ы", "ы", "ы", "ы", "ы", "ы"]

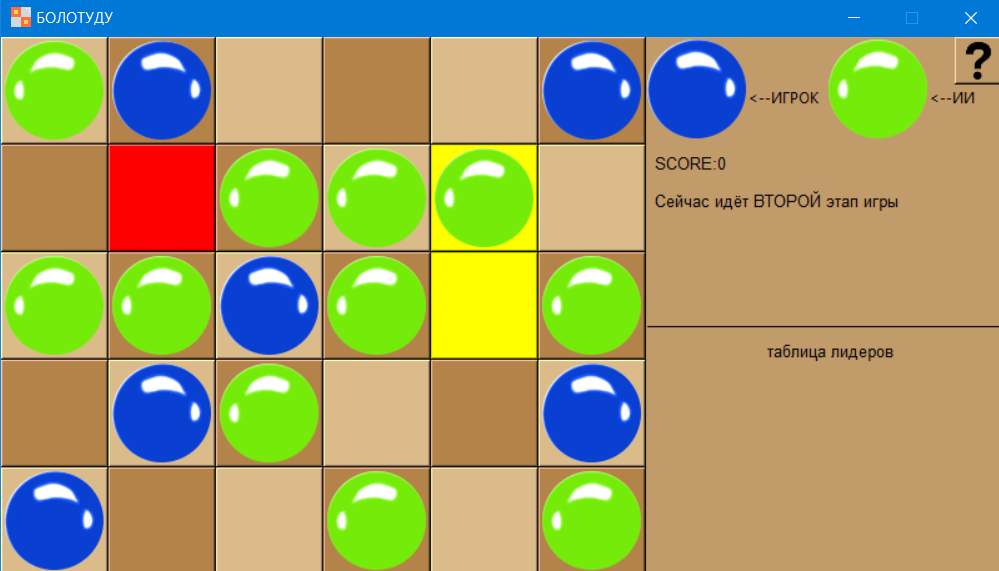
0 – (пустые) клетки свободны, 1 – клетка с фишкой игрока, 2 – клетка с фишкой компьютера, "ы" – мусорная клетка.

Модель поля для игрока

Первый этап

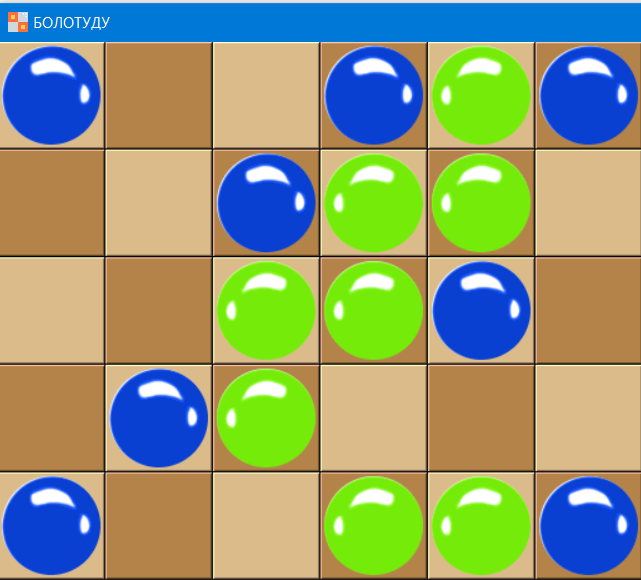


Второй этап



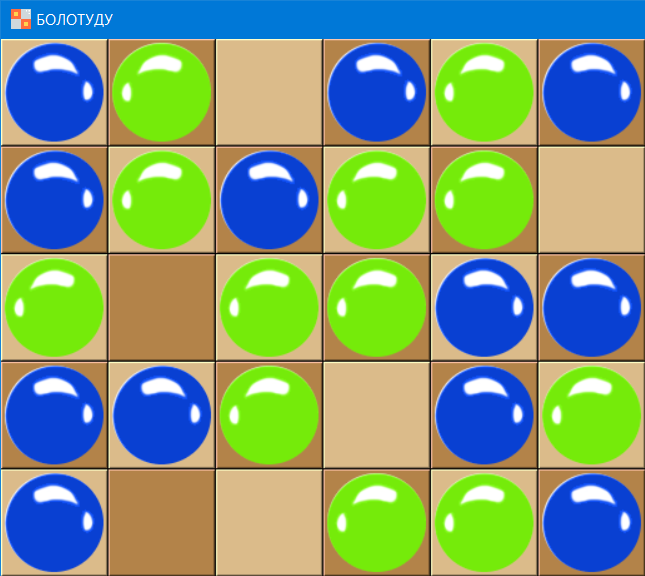
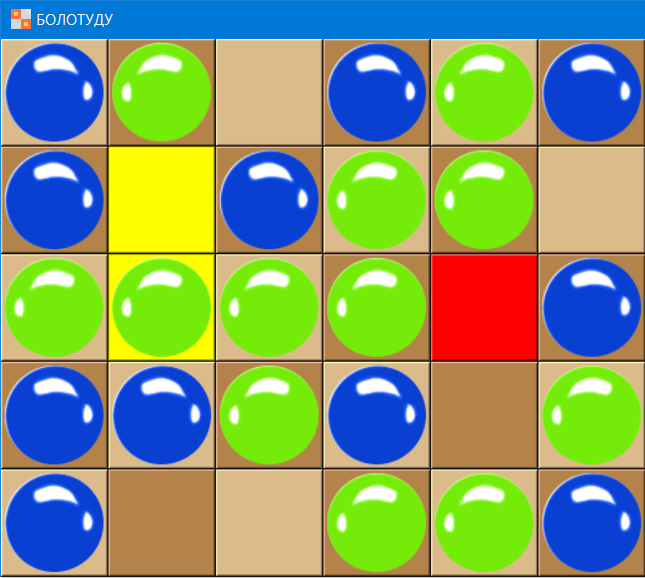
**Модель поведения компьютера в первом этапе**

Компьютер в первом этапе игры будет стараться предотвращать использование возможных троек из фишек игрока. Делает он это ставя фишки на продолжении двух рядом стоящих фишек игрока или находящихся на одной линии через одну клетку (левый рисунок). Если таких возможностей нет: или по правилам игры не получится перекрыть возможную тройку игрока, или игрок не создаст подобную ситуацию, то компьютер будет стараться ставить свои фишки рядом друг с другом, чтобы у него были возможные тройки для второго этапа (правый рисунок). (синие фишки – игрок, зелёные - компьютер)

**Модель поведения компьютера во втором этапе**

Во втором этапе игры компьютер будет атаковать игрока, если такая возможность есть. В данном примере показан возможный вариант игры, в котором компьютер съедает фишку игрока.

**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,

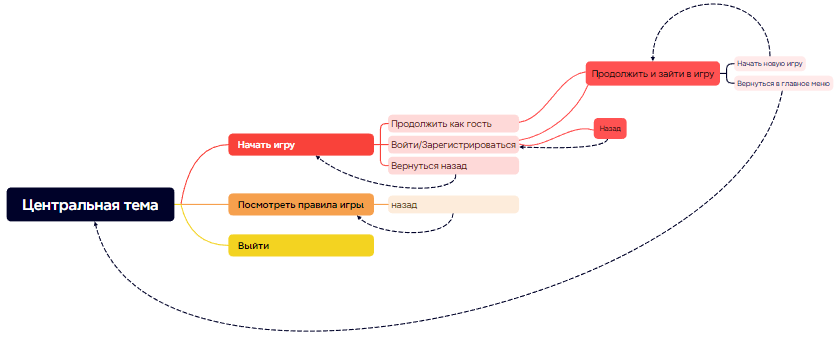
2 этап – передвижение)

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: 3.

Количество обнаруженных и исправленных ошибок в программном

коде: 1.

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 фишек, а также верное выполнение проверки хода игрока

Ожидаемый результат:

Запрет выставления троек в первом этапе игры.

Фактический результат:

Запрет выставления троек в первом этапе игры.

**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: 9.11.23).

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: 05.11.23).

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: 17.12.23).

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 24.10.23)

**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»**

**Тема:** Color Lines

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

**Руководство программиста**

Р.02069337. 22/2359-27 РП-<01>

Листов: 21

Исполнитель:

студентка гр. ИСТбд-22

Афонин Константин Сергеевич

« » 202\_\_ г.

202\_\_ г.

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Программа предназначена для игры в компьютерную игру «Color Lines»

Краткие правила игры:

* Компьютер выставляет три шарика случайных цветов на каждом ходу.
* Игрок может передвинуть любой шарик, если между начальной и конечной клетками есть недиагональный путь.
* Если собирается пять или более шариков одного цвета, они исчезают и игрок получает возможность сделать ещё одно перемещение.
* На экране показываются три цвета шариков, которые будут выброшены на поле на следующем ходу компьютера.

Цель игры – набрать наибольшее количество очков.

**1.2 Условия, необходимые для использования приложения**

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

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: 753.

Количество алгоритмов: 32.

Используемые библиотеки:

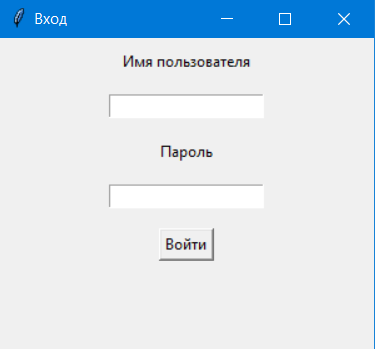
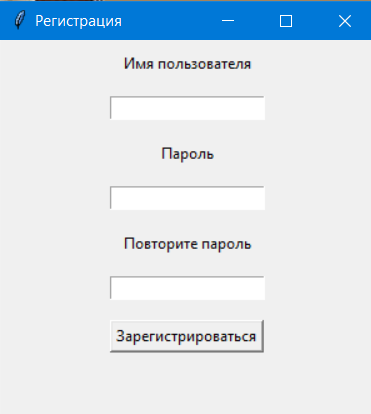
1. tkinter – библиотека для разработки графического интерфейса; в приложении используется для создания графического окна с виджетами;
2. PIL – библиотека языка Python (версии 2), предназначенная для работы с растровой графикой; в приложении библиотека используется для использования изображений библиотекой tkinter;
3. sqlite3 - это встроенная база данных, которая предоставляет мощные возможности хранения и управления данными внутри приложений.
4. random – Этот модуль реализует генераторы псевдослучайных чисел под различные потребности.

Описание приложения:

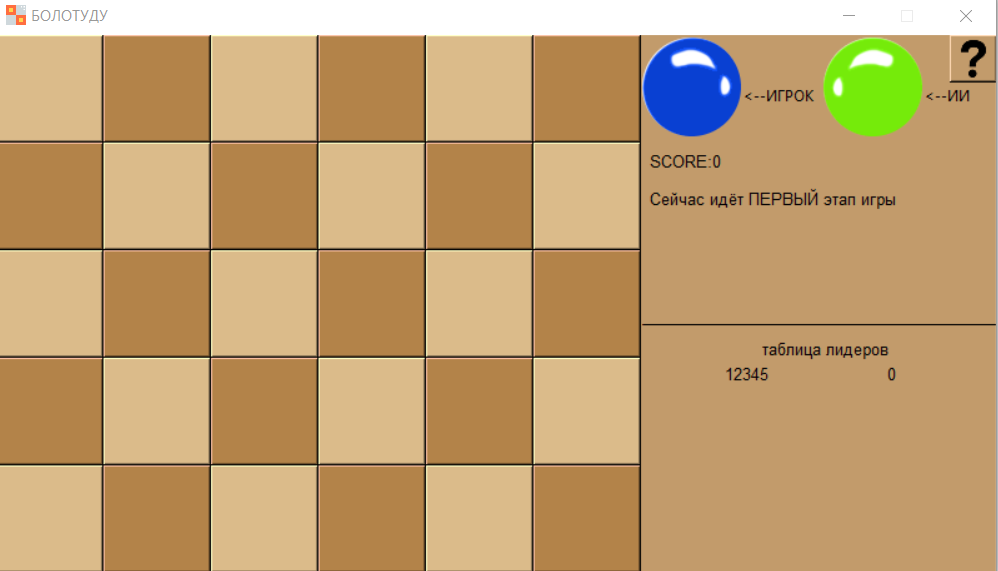
При запуске приложения появляется окно главного меню.

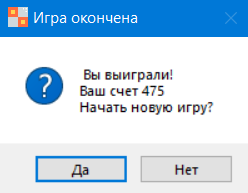
Пользователю предоставляется на выбор три действия: при нажатии на кнопку “Начать игру” откроется окно с выбором регистрации или авторизации, а также кнопка возвращения назад в главное меню; при нажатии на кнопку “Правила игры” пользователь сможет увидеть правила игры; при нажатии на кнопку “Выход” приложение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

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



Справа от игрового поля пользователь видит свой счёт, таблицу лидеров, 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 Bolotudu

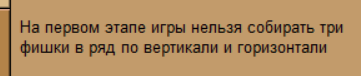
1. \_\_init\_\_ - инициализация;
2. rules – функция отрисовки окна с правилами во время игры;
3. colour\_change – функция перекраски игрового поля
4. change\_part – функция перехода во второй этап игры
5. click – функция обработки хода(клика) игрока
6. choice\_cell – функция обработки хода игрока во втором этапе игры
7. check\_attack – функция проверки: атакует ли игрок/компьютер
8. check\_three\_on\_line – функция проверки на три в ряд
9. pc\_possiblemove – функция поиска всевозможных ходов компьютера
10. pc\_move – функция обработки хода компьютера в первом этапе игры
11. pc\_choice – функция выбора поля компьютером в первом этапе игры
12. pc\_part\_2 – функция обработки хода компьютера во втором этапе игры
13. direction – выбор направления хода компьютера
14. direction\_move – функция отрисовки хода компьютера
15. pc\_attack – функция атаки компьютера
16. win – функция проверки окончания игры
17. change\_table\_of\_records – функция заполнения таблицы лидеров
18. file\_sourt – функция сортировки файла для таблицы лидеров
19. table\_of\_records – функция вывод таблицы лидеров

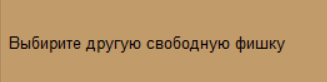
Используемые библиотеки:

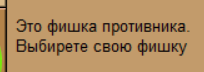
1. tkinter – библиотека для разработки графического интерфейса; в приложении используется для создания графического окна с виджетами;
2. PIL – библиотека языка Python (версии 2), предназначенная для работы с растровой графикой; в приложении библиотека используется для использования изображений библиотекой tkinter;
3. s qlite3 - это встроенная база данных, которая предоставляет мощные возможности хранения и управления данными внутри приложений.
4. random – Этот модуль реализует генераторы псевдослучайных чисел под различные потребности.

**4. Сообщения**

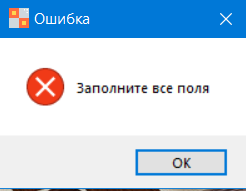
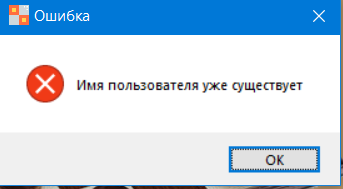
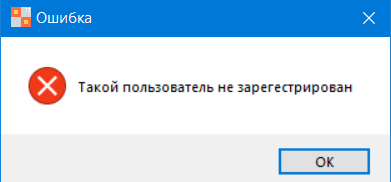
При попытке сходить не в соответствии с правилами игры, пользователь видит сообщение:







Если во время регистрации пользователь ошибается, от видит предупреждения:

**Текст программы:**

from tkinter import \*

import tkinter.messagebox as mb

from PIL import ImageTk

import sqlite3

import random

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="Bolotudu\_Game/icon.png", master=self))

self.title('БОЛОТУДУ')

self.configure(bg="#c29b6b")

w = 800

h = 430

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="Bolotudu\_Game/TheChessboard.png", master=self)

self.lbl\_front = Label(self, image=self.image, bg="#c29b6b", font='Areal 14', justify="left",

wraplength=550)

self.lbl\_front.place(x=0, y=0)

lbl\_name = Label(self, bg="#c29b6b", text="шашечная игра\nБОЛОТУДУ", font='Arial 20')

lbl\_name.place(x=575, y=40)

self.btn\_startgame = Button(self, bg="#c29b6b", text="Начать игру", font="Arial 12", command=self.new\_game)

self.btn\_startgame.place(x=630, y=250)

self.btn\_rule = Button(self, bg="#c29b6b", text="Правила игры", font="Arial 12", command=self.changing\_lbl)

self.btn\_rule.place(x=625, y=290)

self.btn\_out = Button(self, bg="#c29b6b", text="Выход", font="Arial 12", command=lambda: exit(0))

self.btn\_out.place(x=652, y=330)

def changing\_lbl(self): # смена правил игры на иконку шахмат 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('Bolotudu\_Game/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 12", command=

self.start\_game\_without\_user)

self.btn\_enter = Button(self, bg="#c29b6b", text="Войти в профиль", font="Arial 12", command=self.enter)

self.btn\_regist = Button(self, bg="#c29b6b", text="Зарегистрироваться", font="Arial 12",

command=self.register)

self.btn\_back = Button(self, bg="#c29b6b", text="Назад", font="Arial 12", command=self.back)

self.btn\_guest.place(x=595, y=250)

self.btn\_enter.place(x=615, y=290)

self.btn\_regist.place(x=600, y=330)

self.btn\_back.place(x=655, y=370)

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("Bolotudu\_Game/User.txt", 'r+') as f:

f.truncate()

Bolotudu(self)

self.deiconify()

def overwrite\_user(self, username):

with open("Bolotudu\_Game/User.txt", "w+") as f2:

f2.truncate()

f2.write(username)

def CreateNewUser(self, username, password, password\_again): # проверка на правильное заполнение полей в окне рег.

if username == '' or password == '' or password\_again == '':

msg = 'Заполните все поля'

mb.showerror("Ошибка", msg)

else:

try:

f1 = open('Bolotudu\_Game/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()

Bolotudu(self)

self.deiconify()

return

except:

f1 = open('Bolotudu\_Game/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()

Bolotudu(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:

try:

f1 = open('Bolotudu\_Game/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()

Bolotudu(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

except:

msg = 'Еще ни один пользователь не зарегестрирован'

mb.showerror("Ошибка", msg)

self.window\_enter.destroy()

self.register()

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 Bolotudu(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="Bolotudu\_Game/icon.png"))

self.title('БОЛОТУДУ')

self.configure(bg="#c29b6b")

w = 800

h = 430

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.first\_part = True # флаг этапа игры (первый этап - раставление, второй - передвижение)

self.choice = False # флаг выбрана ли фишка во втором этапе

self.score = 0 # очки игрока

self.zeros = [] # массив в который записываются координаты фишек оппонента для выбора: "которую из них съесть"

self.player = 1 # кто сейчас играет: 1 - игрок; 2 - компьютер

self.count = 0 # счетчик фишек для первого этапа игры

self.count\_figur = 0 # счетчик поставленных фишек для перехода во второй этап игры

self.field = [["ы", "ы", "ы", "ы", "ы", "ы", "ы", "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"], # поле игры (ы-мусор, 0-путое поле, Х-игрк, О-комп)

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", 0, 0, 0, 0, 0, 0, "ы"],

["ы", "ы", "ы", "ы", "ы", "ы", "ы", "ы"]]

self.image1 = ImageTk.PhotoImage(file="Bolotudu\_Game/green.png")

self.image2 = ImageTk.PhotoImage(file="Bolotudu\_Game/blue.png")

self.image3 = ImageTk.PhotoImage(file="Bolotudu\_Game/empty.png")

image\_settings = ImageTk.PhotoImage(file="Bolotudu\_Game/question\_mark.png")

self.button\_help = Button(self, bg="#c29b6b", image=image\_settings, command=self.rules)

self.button\_help.place(x=763, y=0)

lbl\_2 = Label(self, bg="#c29b6b", text='<--ИГРОК')

lbl\_1 = Label(self, bg="#c29b6b", text='<--ИИ')

lbl\_2.place(x=595, y=38)

lbl\_1.place(x=740, y=38)

panel1 = Label(self, bg="#c29b6b", image=self.image1)

panel2 = Label(self, bg="#c29b6b", image=self.image2)

panel1.place(x=660, y=0)

panel2.place(x=515, y=0)

self.lbl\_score = Label(self, bg="#c29b6b", text=f'SCORE:{self.score}', font='Areal 10')

self.lbl\_score.place(x=520, y=90)

self.lbl\_part = Label(self, bg="#c29b6b", text="Сейчас идёт ПЕРВЫЙ этап игры", font='Areal 10')

self.lbl\_part.place(x=520, y=120)

self.lbl\_popup = Label(self, bg="#c29b6b", text='', font='Areal 10', justify='left', wraplength=278)

self.lbl\_popup.place(x=520, y=175)

lbl\_line = Label(self, bg="#c29b6b", text='\_'\*57)

lbl\_line.place(x=514, y=215)

lbl\_score = Label(self, bg="#c29b6b", text='таблица лидеров', font='Areal 10')

lbl\_score.place(x=610, y=240)

self.lbl\_table\_1 = Label(self, bg="#c29b6b", text='', font="Areal 10")

self.lbl\_table\_2 = Label(self, bg="#c29b6b", text='', font="Areal 10")

self.lbl\_table\_1.place(x=580, y=260)

self.lbl\_table\_2.place(x=710, y=260)

self.table\_of\_records()

self.b = [] # заготовка поля из матрицы кнопок для визуала

for i in range(7):

self.b.append(i)

self.b[i] = []

for j in range(8):

self.b[i].append(j)

if (0 < i < 6) and (0 < j < 7): # выводим только поле размером 5х6 с сохранением координат 7х8

self.b[i][j] = Button(self, height=80, width=80, image=self.image3,

command=lambda r=i, c=j: self.click(r, c))

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="Bolotudu\_Game/icon.png"))

rule.title("ПРАВИЛА ИГРЫ")

rule.geometry("462x330")

rule.configure(bg="#c29b6b")

rule.resizable(False, False)

with open('Bolotudu\_Game/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, 6):

for j in range(1, 7):

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.win\_X = 12

self.win\_O = 12

self.count\_figur = 0

self.first\_part = False

self.lbl\_part.configure(text='Сейчас идёт ВТОРОЙ этап игры')

return

def click(self, r, c): # обработчик кнопок поля

try:

self.lbl\_popup.configure(text='')

if self.first\_part: # если идёт первый этап игры

self.colour\_change()

if not self.check\_three\_on\_line(r, c):

self.count += 1

self.b[r][c].configure(image=self.image2)

self.field[r][c] = 1

if self.count == 2:

self.count = 0

self.player = 2

self.pc\_move()

else: # если не первый этап игры

self.choice\_cell(r, c)

return

except Exception as e:

pass

def choice\_cell(self, r, c): # второй этап игры

if len(self.zeros) != 0: # если несколько вариантов атаки

for i in self.zeros:

if i == (r, c): # если выбранная игроком фишка противника подходит

self.b[r][c].configure(image=self.image3)

self.field[r][c] = 0

self.colour\_change()

self.score += 50 \* len(self.zeros)

self.lbl\_score.configure(text=f'SCORE:{self.score}')

self.zeros = []

self.win\_X -= 1

self.win()

self.player = 2

self.pc\_move()

break

else:

self.lbl\_popup.configure(text='Выберете выделенную фишку противника')

return

if not (self.choice): # выбор своей фишки для хода ею

if self.field[r][c + 1] != 0 and self.field[r][c - 1] != 0 and self.field[r + 1][c] != 0 \

and self.field[r - 1][c] != 0 and self.field[r][c] == 1:

self.lbl\_popup.configure(text='Выбирите другую свободную фишку')

elif self.field[r][c] == 2:

self.lbl\_popup.configure(text='Это фишка противника.\nВыбирете свою фишку')

elif self.field[r][c] == 1:

self.colour\_change()

self.a, self.z = r, c

self.b[r][c].configure(bg='yellow')

self.choice = True

elif self.a == r and self.z == c: # отменить выбранную фишку

self.choice = False

self.colour\_change()

else: # ход выбранной фишкой

if (abs(r - self.a) + abs(c - self.z) == 1) and self.player == 1 and self.field[r][c] == 0:

self.b[r][c].configure(image=self.image2)

self.field[r][c] = 1

self.b[self.a][self.z].configure(image=self.image3)

self.colour\_change()

self.field[self.a][self.z] = 0

self.zeros = self.check\_attack(r, c)

if len(self.zeros) == 1: # если под атакой 1 вражеская фишка

self.b[self.zeros[0][0]][self.zeros[0][1]].configure(image=self.image3)

self.field[self.zeros[0][0]][self.zeros[0][1]] = 0

self.zeros = []

self.score += 50

self.lbl\_score.configure(text=f'SCORE:{self.score}')

self.win\_X -= 1

self.win()

if len(self.zeros) == 0: # если под атакой нет вражеских фишек

self.player = 2

self.pc\_move()

else: # если под атакой несколько вражеских фишек

for i in self.zeros:

self.b[i[0]][i[1]].configure(bg='red')

self.choice = False

else:

self.b[self.a][self.z].configure(bg='yellow')

self.lbl\_popup.configure(text='Для передвижения фишки выберете соседнее'

' поле по вертикали или горизонтали')

return

def check\_attack(self, r, c): # проверка на атаку

zero\_nymnym = []

cross\_nymnym = []

right = 0

left = 0

for i in range(c, c + 2): # проверка наличия фишек игрока/противника справа

if self.field[r][i + 1] == self.player:

right += 1

else:

break

for i in range(c, c - 2, -1): # проверка наличия фишек игрока/противника слева

if self.field[r][i - 1] == self.player:

left += 1

else:

break

if right + left >= 2 and self.player == 1:

if self.field[r][c + right + 1] == 2:

zero\_nymnym.append((r, c + right + 1))

if self.field[r][c - left - 1] == 2:

zero\_nymnym.append((r, c - left - 1))

elif right + left >= 2 and self.player == 2:

if self.field[r][c + right + 1] == 1:

cross\_nymnym.append((r, c + right + 1))

if self.field[r][c - left - 1] == 1:

cross\_nymnym.append((r, c - left - 1))

up = 0

down = 0

for i in range(r, r + 2): # проверка наличия фишек игрока/противника снизу

if self.field[i + 1][c] == self.player:

down += 1

else:

break

for i in range(r, r - 2, -1): # проверка наличия фишек игрока/противника сверху

if self.field[i - 1][c] == self.player:

up += 1

else:

break

if down + up >= 2 and self.player == 1:

if self.field[r + down + 1][c] == 2:

zero\_nymnym.append((r + down + 1, c))

if self.field[r - up - 1][c] == 2:

zero\_nymnym.append((r - up - 1, c))

elif down + up >= 2 and self.player == 2:

if self.field[r + down + 1][c] == 1:

cross\_nymnym.append((r + down + 1, c))

if self.field[r - up - 1][c] == 1:

cross\_nymnym.append((r - up - 1, c))

if self.player == 1:

return zero\_nymnym

if self.player == 2:

return cross\_nymnym

def check\_three\_on\_line(self, r, c): # проверка на три в ряд в первом этапе игры

if self.field[r][c] == 0 and self.count < 2:

if (self.field[r][c - 2] == self.player and self.field[r][c - 1] == self.player) or \

(self.field[r][c + 1] == self.player and self.field[r][c + 2] == self.player) or \

(self.field[r][c - 1] == self.player and self.field[r][c + 1] == self.player): # по горизонтали

if self.player == 1: # исход для игрока

self.b[r][c].configure(bg="red")

self.lbl\_popup.configure(text='На первом этапе игры нельзя собирать '

'три фишки в ряд по вертикали и горизонтали')

return True

if self.player == 2: # исход для компа

return [r, c]

if (self.field[r + 1][c] == self.player and self.field[r + 2][c] == self.player) or \

(self.field[r - 1][c] == self.player and self.field[r - 2][c] == self.player) or \

(self.field[r - 1][c] == self.player and self.field[r + 1][c] == self.player): # по вертикали

if self.player == 1:

self.b[r][c].configure(bg="red")

self.lbl\_popup.configure(text='На первом этапе игры нельзя собирать '

'три фишки в ряд по вертикали и горизонтали')

return True

if self.player == 2:

return [r, c]

elif self.field[r][c] != 0 and self.player == 1: # если кликнул не на пустое поле

return True

elif self.player == 2:

return

elif self.player == 1: # если поле пустое

return False

def pc\_possiblemove(self): # поиск фишек которыми может сходить комп

possiblemove = []

for i in range(1, 6):

for j in range(1, 7):

if self.field[i][j] == self.player and (self.field[i][j + 1] == 0 or self.field[i][j - 1] == 0 or

self.field[i + 1][j] == 0 or self.field[i - 1][j] == 0):

possiblemove.append([i, j])

return possiblemove

def pc\_move(self): # ход компьютера

if self.first\_part: # если первый этап игры

while self.count < 2:

choice\_field = self.pc\_choice()

if len(choice\_field) != 0: # комп перекрывать фишки игрока => игрок не может сделать тройку

move = random.choice(choice\_field)

self.b[move[0]][move[1]].configure(image=self.image1)

self.field[move[0]][move[1]] = 2

self.count += 1

self.count\_figur += 1

self.change\_part()

else:

if len(self.free\_field) != 0: # комп ставит фишки рядом со своими

move = random.choice(self.free\_field)

self.b[move[0]][move[1]].configure(image=self.image1)

self.field[move[0]][move[1]] = 2

self.count += 1

self.count\_figur += 1

self.change\_part()

else:

if [3, 3] in self.other\_field or [3, 4] in self.other\_field: # если перекрывать нечего и

if [3, 3] in self.other\_field:

field = [[3, 3]] # если нет свободных со своими

if [3, 4] in self.other\_field:

field = [[3, 4]]

move = random.choice(field)

else:

move = random.choice(self.other\_field)

self.b[move[0]][move[1]].configure(image=self.image1)

self.field[move[0]][move[1]] = 2

self.count += 1

self.count\_figur += 1

self.change\_part()

else:

self.player = 1

self.count = 0

return

else:

choice\_usefulmove = self.pc\_part\_2()

if len(choice\_usefulmove) != 0: # если есть ходы, которые заберут фишку игрока

move = random.choice(choice\_usefulmove)

choice\_direction = self.direction(move[0], move[1])

while True:

direction = random.choice(choice\_direction) # выбор направления

self.field[move[0]][move[1]] = 0

if direction == 1 and len(self.check\_attack(move[0], move[1] + 1)) != 0:

self.field[move[0]][move[1]] = 2

break

elif direction == 2 and len(self.check\_attack(move[0] + 1, move[1])) != 0:

self.field[move[0]][move[1]] = 2

break

elif direction == 3 and len(self.check\_attack(move[0], move[1] - 1)) != 0:

self.field[move[0]][move[1]] = 2

break

elif direction == 4 and len(self.check\_attack(move[0] - 1, move[1])) != 0:

self.field[move[0]][move[1]] = 2

break

self.direction\_move(direction, move) # ход компа

if direction == 1:

cross\_nymnym = self.check\_attack(move[0], move[1] + 1)

self.pc\_attack(cross\_nymnym)

return

elif direction == 3:

cross\_nymnym = self.check\_attack(move[0], move[1] - 1)

self.pc\_attack(cross\_nymnym)

return

elif direction == 2:

cross\_nymnym = self.check\_attack(move[0] + 1, move[1])

self.pc\_attack(cross\_nymnym)

return

elif direction == 4:

cross\_nymnym = self.check\_attack(move[0] - 1, move[1])

self.pc\_attack(cross\_nymnym)

return

elif len(self.pc\_possiblemove()) != 0:

possiblemove = self.pc\_possiblemove()

move = random.choice(possiblemove)

direction\_choice = self.direction(move[0], move[1])

direction = random.choice(direction\_choice)

self.direction\_move(direction, move)

self.player = 1

return

else:

self.lbl\_popup.configure(text='Противнику некуда ходить.\nХод противника пропускается')

self.player = 1

return

def pc\_choice(self):

self.free\_field = [] # поля рядом с фишкой компа

self.other\_field = [] # поля не рядом с фишкой компа и без перекрытия фишек игрока

pc\_choise\_field = [] # поля для перекрытия фишек игрока

for i in range(1, 6):

for j in range(1, 7):

if self.field[i][j] == self.field[i][j + 1] == 1:

if self.field[i][j - 1] == 0 and not(self.check\_three\_on\_line(i, j - 1)):

pc\_choise\_field.append([i, j - 1])

if self.field[i][j + 2] == 0 and not(self.check\_three\_on\_line(i, j + 2)):

pc\_choise\_field.append([i, j + 2])

if self.field[i][j] == self.field[i + 1][j] == 1:

if self.field[i - 1][j] == 0 and not(self.check\_three\_on\_line(i - 1, j)):

pc\_choise\_field.append([i - 1, j])

if self.field[i + 2][j] == 0 and not(self.check\_three\_on\_line(i + 2, j)):

pc\_choise\_field.append([i + 2, j])

if self.field[i][j - 1] == 1 and self.field[i][j + 1] == 1 and self.field[i][j] == 0 and\

not(self.check\_three\_on\_line(i, j)):

pc\_choise\_field.append([i, j])

if self.field[i - 1][j] == 1 and self.field[i + 1][j] == 1 and self.field[i][j] == 0 and\

not(self.check\_three\_on\_line(i, j)):

pc\_choise\_field.append([i, j])

for i in range(1, 6):

for j in range(1, 7):

if not([i, j] in pc\_choise\_field) and not(self.check\_three\_on\_line(i, j)) \

and self.field[i][j] == 0 and (self.field[i][j + 1] == 2 or self.field[i][j - 1] == 2

or self.field[i + 1][j] == 2 or self.field[i - 1][j] == 2):

self.free\_field.append([i, j])

elif len(self.free\_field) == 0 and not(self.check\_three\_on\_line(i, j)) and self.field[i][j] == 0:

self.other\_field.append([i, j])

return pc\_choise\_field

def pc\_part\_2(self):

possiblemove = self.pc\_possiblemove() # все возможные ходы

usefulmove = [] # все полезные ходы

left\_move = []

right\_move = []

up\_move = []

down\_move = []

for i in possiblemove: # проверка на полезный ход

if i[1] > 1 and self.field[i[0]][i[1] - 1] == 0:

self.field[i[0]][i[1]] = 0

left\_move = self.check\_attack(i[0], i[1] - 1)

self.field[i[0]][i[1]] = 2

if i[1] < 6 and self.field[i[0]][i[1] + 1] == 0:

self.field[i[0]][i[1]] = 0

right\_move = self.check\_attack(i[0], i[1] + 1)

self.field[i[0]][i[1]] = 2

if i[0] > 1 and self.field[i[0] - 1][i[1]] == 0:

self.field[i[0]][i[1]] = 0

up\_move = self.check\_attack(i[0] - 1, i[1])

self.field[i[0]][i[1]] = 2

if i[0] < 5 and self.field[i[0] + 1][i[1]] == 0:

self.field[i[0]][i[1]] = 0

down\_move = self.check\_attack(i[0] + 1, i[1])

self.field[i[0]][i[1]] = 2

if len(left\_move) != 0 or len(right\_move) != 0 or len(up\_move) != 0 or len(down\_move) != 0:

usefulmove.append([i[0], i[1]])

left\_move = right\_move = up\_move = down\_move = []

return usefulmove

def direction(self, i, j): # нахождение стороны направления хода

choice\_direction = []

if self.field[i][j + 1] == 0:

choice\_direction.append(1)

if self.field[i + 1][j] == 0:

choice\_direction.append(2)

if self.field[i][j - 1] == 0:

choice\_direction.append(3)

if self.field[i - 1][j] == 0:

choice\_direction.append(4)

return choice\_direction

def direction\_move(self, d, move): # ход компа в выбранную сторону выбранной фишкой

self.b[move[0]][move[1]].configure(bg="yellow")

if d == 1:

self.b[move[0]][move[1] + 1].configure(image=self.image1)

self.field[move[0]][move[1] + 1] = 2

self.b[move[0]][move[1] + 1].configure(bg="yellow")

self.b[move[0]][move[1]].configure(image=self.image3)

self.field[move[0]][move[1]] = 0

return

elif d == 3:

self.b[move[0]][move[1] - 1].configure(image=self.image1)

self.field[move[0]][move[1] - 1] = 2

self.b[move[0]][move[1] - 1].configure(bg="yellow")

self.b[move[0]][move[1]].configure(image=self.image3)

self.field[move[0]][move[1]] = 0

return

elif d == 2:

self.b[move[0] + 1][move[1]].configure(image=self.image1)

self.field[move[0] + 1][move[1]] = 2

self.b[move[0] + 1][move[1]].configure(bg="yellow")

self.b[move[0]][move[1]].configure(image=self.image3)

self.field[move[0]][move[1]] = 0

return

elif d == 4:

self.b[move[0] - 1][move[1]].configure(image=self.image1)

self.field[move[0] - 1][move[1]] = 2

self.b[move[0] - 1][move[1]].configure(bg="yellow")

self.b[move[0]][move[1]].configure(image=self.image3)

self.field[move[0]][move[1]] = 0

return

def pc\_attack(self, cross\_nymnym):

cross = random.choice(cross\_nymnym)

self.b[cross[0]][cross[1]].configure(image=self.image3)

self.b[cross[0]][cross[1]].configure(bg='red')

self.field[cross[0]][cross[1]] = 0

if self.score >= 25:

self.score -= 25

self.lbl\_score.configure(text=f'SCORE:{self.score}')

self.win\_O -= 1

self.win()

self.player = 1

return

def win(self): # условия победы

if self.win\_X == 2:

self.change\_table\_of\_records()

self.file\_sourt()

self.table\_of\_records()

win = mb.askquestion('Игра окончена', f' Вы выиграли!\n Ваш счет {self.score}\n Начать новую игру?')

if win == 'yes':

self.lbl\_score.configure(text='0')

self.reload()

else:

self.destroy()

elif self.win\_O == 2:

self.change\_table\_of\_records()

self.file\_sourt()

self.table\_of\_records()

win = mb.askquestion('Игра окончена', f'Компьютер выиграл\n Ваш счет {self.score}\n Начать новую игру?')

if win == 'yes':

self.lbl\_score.configure(text='0')

self.reload()

else:

self.destroy()

def change\_table\_of\_records(self): # добавление нового результата в таблицу лидеров

f2 = open("Bolotudu\_Game/User.txt", "r")

text2 = f2.read()

if len(text2) != 0:

f1 = open("Bolotudu\_Game/Users.txt", "r+")

text1 = f1.read().split()

new\_text = []

for i in text1:

if i.split(":")[0] == text2 and int(i.split(":")[2]) < self.score:

new\_record = i.rstrip(i.split(":")[2]) + str(self.score)

new\_text.append(new\_record)

else:

new\_text.append(i)

f1.seek(0)

f1.truncate()

f1.write('\n'.join(new\_text) + '\n')

f1.close()

f2.close()

return

def file\_sourt(self): # сортировка лидеров

f = open('Bolotudu\_Game/Users.txt', 'r+')

text = f.read().split()

sorted\_text = sorted(text, key=lambda x: -int(x.split(':')[2]))

sorted\_text\_str = "\n".join(sorted\_text)

f.seek(0)

f.write(sorted\_text\_str)

f.close()

return

def table\_of\_records(self): # вывод таблици лидеров

f = open('Bolotudu\_Game/Users.txt', 'r')

text = f.read().split()

name\_users = [i.split(':')[0] for i in text]

score\_users = [i.split(':')[2] for i in text]

name\_users\_str = "\n".join(name\_users)

score\_users\_str = "\n".join(score\_users)

self.lbl\_table\_1.configure(text=name\_users\_str)

self.lbl\_table\_2.configure(text=score\_users\_str)

f.close()

return

menu = Menu()

menu.mainloop()