МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» Кафедра «Измерительно-вычислительные комплексы»

Дисциплина

«Алгоритмы и структуры данных»

09.03.02 Информационные системы и технологии

Курсовая работа

Разработка компьютерной логической игры «Крестики-нолики до пяти в ряд»

Руководитель разработки:

Кандидат технических наук,

доцент кафедры ИВК

Шишкин Вадим Викторинович

« » 2024г.

Исполнитель: студент гр. ИСТбд-21

*Васильев Дмитрий Сергеевич*

« » 2024 г.

2024

**СОДЕРЖАНИЕ**

[**Аннотация** 3](#_Toc185934296)

[**ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 4](#_Toc185934297)

[**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА** 8](#_Toc185934298)

[**РУКОВОДСТВО ПРОГРАММИСТА** 23](#_Toc185934299)

[**Заключение** 29](#_Toc185934300)

[**Список использованных источников** 30](#_Toc185934301)

[**Текст программы** 32](#_Toc185934302)

[**ПРИЛОЖЕНИЕ 1** 55](#_Toc185934303)

# **Аннотация**

Курсовая работа посвящена разработке компьютерной логической игры «Крестики-нолики до пяти в ряд». Основной целью работы являлось создание игрового приложения, способного поддерживать два режима игры: между двумя пользователями и против компьютерного бота. Приложение реализует классические правила игры с добавлением функциональных возможностей, таких как выбор режима игры, подсчет времени и ходов, а также отображение статистики. Одной из приоритетных задач проекта было разработка компьютерного бота, использующего алгоритм Minimax с оценочной фукнцией позиций, что позволило значительно повысить уровень сложности игры.

Разработка включала создание графического интерфейса на языке программирования Python с использованием библиотеки Tkinter и внедрение логики игры на основе классов и структур. Были изучены математические модели и алгоритмы, необходимые для реализации игры, проведено тестирование различных сценариев работы приложения. Также была разработана техническое задание, подробная пояснительная записка и руководство программиста.

Полученное приложение демонстрирует высокую степень соответствия требованиям технического задания и обладает широкими возможностями для дальнейшего совершенствования и расширения функционала. Особенно стоит отметить успешную реализацию компьютерного бота, который стал центральным элементом игры, делая ее интересной и увлекательной для игроков разного уровня подготовки.

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

|  |  |
| --- | --- |
| **Подп. и** |  |
| **Инв** |  |
| **Вза** |  |
| **Подп. и** |  |
| **Инв.** |  |

по дисциплине «Алгоритмы и структуры данных»

Тема «Крестики-нолики до пяти в ряд»

Руководитель разработки:

Кандидат технических наук,

доцент кафедры ИВК

Шишкин Вадим Викторинович

« » 2024г.

Исполнитель: студент гр. ИСТбд-21

*Васильев Дмитрий Сергеевич*

« » 2024 г.

2024

**Введение**

Приложение «Крестики-Нолики до пяти в ряд» представляет собой компьютерную версию классической игры с расширенными правилами, где целью является выстроить пять одинаковых знаков подряд. В отличие от традиционного варианта игры, в данной версии поле может быть любого размера, и для победы требуется выстроить линию из пяти символов.

Целью разработки является создание игрового приложения, которое поддерживает два режима: игра между двумя пользователями и игра против компьютера. Приложение реализует классические правила с добавлением новых функциональных возможностей, таких как выбор режима игры, подсчет времени и ходов, а также отображение статистики. В результате, игроки получают возможность как соревноваться с реальными противниками, так и тренироваться в игре против компьютера.

Разработка приложения включает создание интерфейса с использованием библиотеки Tkinter для графического представления игрового поля, а также внедрение логики игры на основе классов и структур, обеспечивающих правильную обработку ходов и определение победителя.

**1. Основания для разработки**

Разработка осуществляется на основании учебного плана дисциплины «Алгоритмы и структуры данных» направления подготовки 09.03.02 «Информационные системы и технологии» и распоряжение по факультету.

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

**2.1. Функциональное назначение**

1. Загрузка и визуализация окна приложения с графическим интерфейсом.
2. Отображение игрового поля с фигурами игрока и противника.
3. Обеспечение интерактивного пользовательского интерфейса для размещения фигур на игровом поле.
4. Определение победителя или ничьей в игре.
5. Отслеживание счета между игроками.
6. Реализация авторизации пользователей с возможностью регистрации.
7. Сохранение и загрузка текущего состояния игры в файл

**2.2. Требования к функциональным характеристикам**

**2.2.1 Требования к структуре приложения**

Приложение должно состоять из нескольких модулей:

1. Модуль графического интерфейса для отображения игрового поля, авторизации пользователей и взаимодействия с ними.
2. Логический модуль, отвечающий за правила игры, проверку условий победы и учет ходов игроков.
3. Модуль авторизации пользователей, включая функции регистрации, входа и выхода из аккаунта.
4. Модуль сохранения и загрузки состояния игры, а также данных пользователей

**2.2.2 Требования к составу функций приложения**

Приложение должно включать следующие функции:

1. Выбор режима игры (игрок против игрока, игрок против компьютера).
2. Интерфейс для начала новой игры и отображения текущего состояния игрового поля.
3. Обработка ходов игроков, проверка правильности ходов и отрисовка фигур на игровом поле.
4. Проверка условий окончания игры и определение победителя.
5. Сохранение и загрузка состояния игры и данных пользователей из файла.
6. Функционал авторизации, включая регистрацию, вход и выход.

**2.2.3 Требования к организации информационного обеспечения, входных и выходных данных**

Приложение должно иметь удобный графический интерфейс, обеспечивающий интуитивное взаимодействие. Игровые данные сохраняются в файлы для возможности восстановления игры и сохранения данных авторизации.

**2.3. Требования к надежности**

Программа должна быть устойчива к ошибкам пользователя. В случае ошибки при вводе данных или выборе неверных действий, программа должна выдавать соответствующие сообщения.

В случае сбоя программы или выхода из приложения пользователи должны иметь возможность восстановить последнее сохраненное состояние игры и продолжить с того места, на котором они остановились.

Все пароли пользователей должны храниться в зашифрованном виде с использованием надежных методов хеширования.

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

Операционная система: Windows;

Используемые технологии: Python 3.8, библиотека Tkinter;

Среда разработки: PyCharm.

**2.5. Требования к маркировке и упаковке**

Определяются заданием на курсовую работу.

**2.6. Требования к транспортированию и хранению**

**2.6.1 Условия транспортирования**

Требования к условиям транспортирования не предъявляются.

**2.6.2 Условия хранения**

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

**2.6.3 Сроки хранения**

Срок хранения – до окончания срока учебы.

**3. Требования к программной документации**

Определяются заданием на курсовую работу.

**4. Стадии и этапы разработки**

Определяются заданием на курсовую работу.

**5. Порядок контроля и приемки**

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»

# **ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

на курсовую работу

|  |  |
| --- | --- |
| **Подп. и** |  |
| **Инв** |  |
| **Вза** |  |
| **Подп. и** |  |
| **Инв.** |  |

по дисциплине «Алгоритмы и структуры данных»

Тема «Крестики-нолики до пяти в ряд»

Руководитель разработки:

Кандидат технических наук,

доцент кафедры ИВК

Шишкин Вадим Викторинович

« » 2024г.

Исполнитель: студент гр. ИСТбд-21

*Васильев Дмитрий Сергеевич*

« » 2024 г.

2024

**Введение**

Приложение «Крестики-Нолики до 5 в ряд» представляет собой компьютерную версию популярной игры, основанной на классических правилах «крестиков-ноликов». Целью разработки данного программного обеспечения является создание симулятора игры «Крестики-Нолики до 5 в ряд», который позволяет пользователям играть как друг против друга, так и против компьютера. Программа предназначена для организации матчей с возможностью выбора размеров игрового поля и сохранения текущего состояния игры.

Основой для разработки послужил учебный план дисциплины «Алгоритмы и структуры данных» направления подготовки 09.03.02 «Информационные системы и технологии», а также распоряжение по факультету. Разработанное приложение включает несколько ключевых модулей: модуль графического интерфейса для отображения игрового поля и взаимодействия с пользователем, логический модуль, отвечающий за правила игры и проверку условий победы, а также модуль взаимодействия с пользователем для выбора режима игры и сохранения состояния игры.

Приложение предлагает удобный графический интерфейс, позволяющий интуитивно взаимодействовать с игрой. Интерфейс был реализован с использованием библиотеки Tkinter, что позволяет создавать графику и элементы управления, такие как кнопки, меню, и таймер. Игра поддерживает возможность сохранения и загрузки текущего состояния, обеспечивая удобство в случае необходимости продолжить игру позже.

Особое внимание уделено обеспечению безопасности. Программа включает функциональность для регистрации и авторизации пользователей. Для защиты паролей применяется шифрование sha256, что гарантирует сохранность данных пользователей. Система также позволяет игрокам выбирать размер игрового поля, что расширяет функциональные возможности приложения.

Все эти элементы делают программу полноценным и функциональным инструментом для игры в «Крестики-Нолики до 5 в ряд», а также удобным и безопасным решением для пользователей.

**1. Проектная часть**

**1.1. Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу.

Детализируется в разработанном техническом задании.

**1.2. Математические методы**

**1.2.1. Модель приложения**

Модель приложения включает описание структуры данных, алгоритмов, а также их взаимосвязь.

**1.2.2. Структуры данных**

В основе игрового поля лежит двухмерная матрица:

где:

−1 — клетка, занятая ноликом;

1 — клетка, занятая крестиком;

0 — пустая клетка.

Размер матрицы (NxN) может быть задан пользователем перед началом игры. По умолчанию размер составляет 15x15, что обеспечивает баланс между сложностью игры и производительностью программы.

Для упрощения проверки игровых условий введено множество направлений для анализа:

Каждое направление представляет вектор, вдоль которого проверяется наличие последовательных фигур.

Также используется дополнительная структура данных для хранения веса клеток, помогающая алгоритму искусственного интеллекта оценивать выгодность ходов. Весовые значения задаются статически и рассчитываются на основе текущего состояния игрового поля.

**1.2.3. Алгоритмы**

Алгоритмы играют ключевую роль в обеспечении логики игры и интеллектуального поведения компьютера.

**1.2.4. Проверка победы**

Алгоритм проверки победы анализирует каждую клетку игрового поля на наличие последовательности из пяти одинаковых символов вдоль заданных направлений. Проверка выполняется итеративно, чтобы гарантировать, что выигрывающая комбинация найдена при минимальных вычислительных затратах. Для каждой клетки проверяются соседние клетки в направлениях из множества directions. Условием победы является выполнение равенства:

где count — количество последовательных фигур одного игрока.

**1.2.5. Алгоритм Minimax**

Для хода компьютера используется алгоритм Minimax с ограничением глубины поиска для оптимизации производительности. Алгоритм анализирует возможные ходы и оценивает их с помощью эвристической функции. Основные параметры функции:

consecutive — длина последовательности фигур в линии;

open\_ends — количество открытых концов последовательности.

Эвристическая функция возвращает значение на основе следующих правил:

Функция f(consecutive,open\_ends) предназначена для оценки стратегической ценности последовательностей фигур на игровом поле. Она адаптируется в зависимости от текущего игрока и учитывает комбинации длины последовательности с количеством открытых концов.

Функция представлена в виде словаря оценок:

**1.2.6. Генерация ходов**

Алгоритм генерации ходов сосредотачивается на клетках, находящихся вблизи уже занятых. Это позволяет сократить количество возможных ходов и ускорить выполнение программы. Проверка наличия соседей вокруг фигуры осуществляется по всем клеткам в радиусе 1.

**1.2.7. Взаимодействие алгоритмов**

Связь между описанными структурами данных и алгоритмами обеспечивает корректную работу приложения. После каждого хода игрока выполняются следующие этапы:

Проверка завершения игры (победа или ничья).

В случае режима игры с компьютером — расчет хода компьютера с помощью алгоритма Minimax.

Обновление состояния игрового поля и интерфейса.

Эта последовательность действий повторяется до окончания матча, что делает игру динамичной и понятной для пользователя.

**1.3. Архитектура и алгоритмы**

**1.3.1 Архитектура приложения**

Разработка архитектуры приложения является важнейшим этапом при создании программного обеспечения. Архитектура позволяет упорядочить взаимодействие программных модулей, определить структуру программы, её поведение и взаимосвязи. Для приложения «Крестики-Нолики до пяти в ряд» архитектура включает в себя описание основных модулей, их функции и взаимодействие. Она разработана с учетом требований из ТЗ, что обеспечивает простоту расширения функционала, модификации и масштабируемости.

Архитектура приложения представляет собой взаимосвязанную совокупность программных и информационных компонентов, распределенных по модулям, каждый из которых отвечает за выполнение конкретных задач. Для приложения «Крестики-Нолики до пяти в ряд» архитектура спроектирована с разделением на ключевые модули.

**1.3.2 Основные модули**

**Логический модуль**

Логический модуль отвечает за реализацию правил игры, обработку ходов, проверку условий победы и работу алгоритма искусственного интеллекта.

**Функционал**:

**Управление игровым полем**: Матрица игрового поля board[i][j], где I, j — индексы строки и столбца, а значения могут быть −1 (нолик), 1 (крестик) или 0 (пустая клетка).

**Проверка победы**: Функция check\_winner(player) проверяет, есть ли у игрока последовательность из пяти фигур в ряду.

**Обработка ходов**: Функция make\_move(row, col) проверяет корректность хода и обновляет состояние поля.

**Ход компьютера**: Алгоритм Minimax (minimax(depth, alpha, beta, maximizing\_player)) используется для выбора оптимального хода с учетом текущего состояния поля.

**Особенности**:

Интеграция с графическим модулем для визуализации изменений.

Динамическая оценка стратегических позиций на поле (функции evaluate\_position() и get\_best\_move()).

**Модуль графического интерфейса (TicTacToeApp)**

Этот модуль реализует визуализацию игрового процесса и взаимодействие с пользователем с использованием библиотеки Tkinter.

**Функционал**:

**Отображение игрового поля**: Поле создается как сетка с помощью виджета Canvas. Для крестиков и ноликов используются графические примитивы (линии и овалы).

**Обработка событий**: Реализована логика кликов мыши для выполнения хода (on\_click()).

**Вывод текущего состояния игры**: Отображается информация о текущем ходе, победителе или ничьей.

**Особенности**:

Интеграция с логическим модулем для обновления игрового поля после каждого хода.

Возможность выбора режима игры через меню управления.

**Модуль управления состоянием игры**

Этот модуль позволяет сохранять и загружать состояние игры для продолжения с прерванного момента.

**Функционал**:

Сохранение текущего состояния игрового поля в файл.

Загрузка состояния из файла для восстановления прерванной игры.

Обеспечение целостности данных при сбое.

**Модуль авторизации пользователей**

Этот модуль отвечает за управление учетными записями пользователей, включая регистрацию, вход и выход из аккаунта. Он реализует механизмы безопасности, такие как хеширование паролей, чтобы обеспечить защиту данных пользователей.  
**Функционал:**

* **Регистрация пользователей:** Позволяет новым пользователям создать учетную запись, проверяя уникальность имени пользователя и совпадение паролей.
* **Вход в систему:** Пользователи могут авторизоваться, вводя имя пользователя и пароль. Пароль проверяется с использованием хеширования.
* **Выход из аккаунта:** Возможность выхода из учетной записи.
* **Хеширование паролей:** Используется алгоритм SHA-256 для защиты паролей пользователей.

**Особенности:**

* Безопасность данных пользователей благодаря хешированию паролей.
* Сохранение и загрузка данных пользователей из файла JSON.

**1.3.4 Взаимодействие модулей**

Модули приложения связаны через определенные интерфейсы:

**Интерактивность**:

Модуль интерфейса принимает события (например, щелчки мыши) и вызывает соответствующие методы логического модуля.

**Управление игровым процессом**:

Логический модуль обновляет состояние игрового поля, которое затем передается в графический интерфейс для отрисовки.

**Работа с данными**:

При необходимости данные игрового поля передаются в модуль управления состоянием для сохранения.

Пример взаимодействия:

1. Пользователь нажимает на клетку игрового поля.
2. Модуль интерфейса вызывает make\_move() логического модуля.
3. Логический модуль проверяет корректность хода, обновляет поле и возвращает результат.
4. Модуль интерфейса перерисовывает игровое поле и выводит текущий статус.

**1.3.5. Особенности и преимущества архитектуры**

**Модульность**: Четкое разделение на функциональные блоки облегчает тестирование, отладку и дальнейшую доработку.

**Гибкость**: Возможность расширения функционала (например, добавление сетевой игры).

**Надежность**: Использование структурированных данных и проверенных алгоритмов минимизирует ошибки.

**Интуитивность**: Простая интеграция пользовательского интерфейса с логической частью делает приложение удобным для пользователя.

Архитектура приложения «Крестики-Нолики до пяти в ряд» использует модульный подход, эффективные алгоритмы и структуры данных, а также их удобство взаимодействия. Она обеспечивает стабильную работу программы и простоту её дальнейшего сопровождения и расширения.

**1.4. Тестирование**

Перед тем как приступить непосредственно к процессу тестирования, крайне важно точно обозначить те аспекты системы, которые подлежат проверке. С этой целью может быть весьма полезным использование интеллектуальных карт, также известных как Mindmap. Интеллектуальная карта представляет собой визуальную схему, с помощью которой можно наглядно отобразить основные функциональные возможности и сценарии использования приложения. Рекомендуется создать такую карту до начала тестирования, что позволит учесть все ключевые элементы (приложение 1).

**1.4.1 Описание отчета о тестировании.**

Отчет о тестировании представляет собой документ, который содержит информацию о проведенных тестах программного обеспечения. Он включает в себя цели тестирования, методику проведения тестов, сами тестовые случаи, а также результаты выполнения этих тестов. Отчет позволяет оценить качество разработанного продукта, выявить возможные ошибки и недочеты, а также определить области, требующие улучшения.

**1.4.2 Цель тестирования**

Целью тестирования является проверка функциональных возможностей приложения «Крестики-нолики до 5 в ряд». Необходимо убедиться, что все компоненты системы работают корректно, интерфейс удобен для пользователей, а программа соответствует предъявленным требованиям. Особое внимание уделяется проверке правильности логики игры, работе пользовательской аутентификации, а также устойчивости приложения к различным сценариям использования

**1.4.3 Методика тестирования**

Для тестирования приложения были разработаны тест-кейсы, каждый из которых описывает определенный сценарий поведения программы. Эти сценарии охватывали различные аспекты работы приложения, включая регистрацию и авторизацию пользователей, игровой процесс, работу таймера и статистику игр, а также взаимодействие с режимами игры. Каждый тест-кейс содержал набор шагов, которые необходимо было выполнить, и ожидаемых результатов, чтобы проверить правильность функционирования программы.

Кроме того, для повышения эффективности тестирования и обеспечения качества программного обеспечения были реализованы автоматизированные тесты с использованием подхода unit-тестирования. Unit-тесты предназначены для проверки отдельных функциональных блоков приложения, таких как методы логического модуля игры, обработка ходов и проверка победителя, а также взаимодействие с компонентами пользовательского интерфейса. Эти тесты позволяют быстро выявить ошибки в коде на ранних этапах разработки и снизить вероятность их появления в дальнейшем.

**1.4.4 Проведенные тесты**

**1.4.4.1 Тест-кейс тестирование**

**Тест-кейс 1: Регистрация пользователя**

**Номер**: TC-001

**Название**: Успешная регистрация нового пользователя

**Предварительные условия**: Открыто окно регистрации

**Шаги**:

1. Ввести имя пользователя "TestUser"
2. Ввести пароль "password123"
3. Подтвердить пароль "password123"
4. Нажать кнопку "Зарегистрироваться"

**Ожидаемый результат**:

1. Появляется сообщение об успешной регистрации
2. Возврат к окну входа
3. Возможность войти с новыми учетными данными

**Тест-кейс 2: Вход в систему**

**Номер**: TC-002

**Название**: Успешная авторизация зарегистрированного пользователя

**Предварительные условия**: Пользователь "TestUser" предварительно зарегистрирован

**Шаги**:

1. Ввести имя пользователя "TestUser"
2. Ввести пароль "password123"
3. Нажать кнопку "Войти"

**Ожидаемый результат**:

1. Успешный вход в игру
2. Отображение имени пользователя в интерфейсе
3. Активное игровое поле

**Тест-кейс 3: Игра против компьютера**

**Номер**: TC-003

**Название**: Полный цикл игры против компьютера

**Предварительные условия**: Пользователь авторизован, выбран режим "Против компьютера"

**Шаги**:

1. Кликнуть на клетку на игровом поле
2. Дождаться хода компьютера
3. Продолжать игру до победы или ничьей

**Ожидаемый результат**:

1. Корректное отображение ходов
2. Попеременные ходы игрока и компьютера
3. Корректное определение победителя или ничьей

**Тест-кейс 4: Смена режима игры**

**Номер**: TC-004

**Название**: Переключение между режимами игры

**Предварительные условия**: Пользователь авторизован

**Шаги**:

1. Изменить режим игры с "Против компьютера" на "Против игрока"
2. Начать новую игру
3. Сделать несколько ходов

**Ожидаемый результат**:

1. Корректное переключение режима
2. Сброс игрового поля
3. Возможность делать ходы в новом режиме

**Тест-кейс 5: Таймер и статистика игры**

**Номер**: TC-005

**Название**: Проверка работы таймера и статистики

**Предварительные условия**: Пользователь в игре

**Шаги**:

1. Начать новую игру
2. Сделать несколько ходов
3. Проверить показания таймера и количества ходов

**Ожидаемый результат**:

1. Таймер корректно отсчитывает время
2. Счетчик ходов увеличивается

**1.4.4.2 Чек-лист тестирования**

**Авторизация и регистрация**

* Регистрация с корректными данными
* Попытка регистрации с существующим именем пользователя
* Вход с неверным паролем
* Вход с неверным именем пользователя

**Игровой процесс**

* Ход игрока в пустую клетку
* Попытка хода в занятую клетку
* Победа крестиков
* Победа ноликов
* Ничья
* Корректность работы режима против компьютера
* Корректность работы режима против игрока

**Интерфейс и дополнительные функции**

* Корректность отображения таймера
* Корректность подсчета ходов
* Функция «Новая игра»
* Выход из учетной записи
* Смена режима игры

**1.4.4.3 Unit-тестирование**

В рамках разработки приложения для игры «Крестики-Нолики до 5 в ряд» было проведено автоматизированное тестирование с использованием unit-тестов. Эти тесты были созданы для проверки ключевых компонентов программы и обеспечения ее надежности. Для тестирования использовался фреймворк unittest, который позволяет проводить как индивидуальные проверки отдельных функций, так и комплексные тесты для всех важных сценариев работы приложения.

**1.4.4.3.1 Тесты авторизации**

Тестирование модуля авторизации включает проверку функциональности регистрации и входа пользователей, а также хеширования паролей. В тестах проверяются следующие аспекты:

1. Регистрация нового пользователя, включая проверку совпадения паролей и регистрацию уже существующего пользователя.
2. Вход в систему с правильными и неправильными данными пользователя.
3. Проверка правильности хеширования паролей и верификации паролей с использованием хеширования.

**1.4.4.3.2 Тесты модуля логики игры**

Для логики игры были созданы тесты, проверяющие различные аспекты игрового процесса:

1. Инициализация игры: проверяется создание игры в разных режимах (игрок против компьютера, игрок против игрока), корректность начальных значений (размер поля, текущий игрок, выигрышная длина).
2. Ходы игроков: проверяется правильность обработки ходов (в том числе в режиме против компьютера), правильность смены игроков и блокировка ходов в занятую клетку.
3. Проверка победных условий: тестируется корректность определения победителя по горизонтали, вертикали, диагонали и обратной диагонали.
4. Завершение игры: проверяется корректное завершение игры после достижения победных условий.
5. Стратегия компьютера: тестируется блокировка победных ходов игроком, что позволяет оценить стратегические решения, принимаемые компьютером.
6. Получение возможных ходов: проверяется правильность получения списка возможных ходов в различные моменты игры.

**1.4.4.3.3 Важные тестовые функции:**

1. test\_password\_hashing: тестирует корректность хеширования паролей и верификацию.
2. test\_computer\_strategy: проверяет, блокирует ли компьютер победные ходы игрока.
3. test\_get\_valid\_moves: тестирует правильность получения доступных ходов в начале игры и после выполнения ходов игроком.

**1.4.5 Выводы**

В результате проведённого тестирования было установлено, что приложение «Крестики-нолики до 5 в ряд» полностью соответствует всем предъявленным требованиям. Все протестированные функции работали корректно, без возникновения сбоев или ошибок.

Исходя из этого, можно сделать вывод о готовности приложения к полноценному использованию. Кроме того, оно имеет значительный потенциал для дальнейшего улучшения и развития функций.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ» Кафедра «Измерительно-вычислительные комплексы»

# **РУКОВОДСТВО ПРОГРАММИСТА**

на курсовую работу

|  |  |
| --- | --- |
| **Подп. и** |  |
| **Инв** |  |
| **Вза** |  |
| **Подп. и** |  |
| **Инв.** |  |

по дисциплине «Алгоритмы и структуры данных»

Тема «Крестики-нолики до пяти в ряд»

Руководитель разработки:

Кандидат технических наук,

доцент кафедры ИВК

Шишкин Вадим Викторинович

« » 2024г.

Исполнитель: студент гр. ИСТбд-21

*Васильев Дмитрий Сергеевич*

« » 2024 г.

2024

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Приложение «Крестики-Нолики до 5 в ряд» предназначено для организации игры между двумя участниками или между игроком и компьютером. Приложение реализует основные функции классической игры «Крестики-Нолики», но с увеличением размера игрового поля и длины выигрышной линии. Кроме того, игра поддерживает дополнительные функции для управления состоянием игры и авторизации пользователей. Основные задачи приложения включают:

1. **Загрузка и визуализация окна игры**: Программа предоставляет графический интерфейс для взаимодействия пользователей, включая отображение игрового окна и элементов управления.
2. **Отображение игрового поля**: Игровое поле отображается на экране и может быть настроено с различными размерами в зависимости от выбора пользователя. На поле размещаются крестики и нолики двух игроков или игрока и компьютера.
3. **Интерактивный пользовательский интерфейс**: Приложение предлагает интуитивно понятный интерфейс, который позволяет игрокам с легкостью размещать свои фигуры на игровом поле. Взаимодействие с игрой происходит с помощью кликов мыши.
4. **Определение победителя или ничьей**: После завершения партии программа автоматически проверяет условия победы или фиксирует ничью, если ни один из игроков не достиг выигрышной комбинации.
5. **Ведение счёта сыгранных партий**: Приложение отслеживает статистику игр, включая количество побед, поражений и ничьих для каждого игрока, а также сохраняет результаты после каждой игры.
6. **Сохранение и загрузка игры**: Программа предоставляет пользователям возможность сохранять текущее состояние игры в файл для последующего продолжения, а также загружать сохраненные игры.
7. **Авторизация пользователей**: Для удобства сохранения статистики и персонализированного игрового процесса приложение поддерживает регистрацию и авторизацию пользователей. Пароли пользователей шифруются для обеспечения безопасности данных.

Эти функции делают приложение удобным и функциональным инструментом для игры в «Крестики-Нолики до 5 в ряд» с возможностью личной статистики и продолжения игры в любое время.

**1.2 Условия, необходимые для использования приложения**

Для корректной работы приложения требуются следующие компоненты:

**Операционная система**: Windows (с установленным Python).

**Python**: Версия 3.8 или выше.

**Библиотеки Python**:

tkinter: для создания графического интерфейса пользователя.

**Среда разработки** (рекомендуется): PyCharm или любая другая IDE, поддерживающая Python, для упрощения процесса написания и отладки кода.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Приложение «Крестики-Нолики до 5 в ряд» реализовано с использованием модульной архитектуры, что обеспечивает гибкость и удобство при дальнейших модификациях и расширении функционала. Программа состоит из нескольких основных модулей, каждый из которых выполняет специфические функции:

1. **Модуль графического интерфейса**: Этот модуль отвечает за создание и управление окном игры, отображение игрового поля, а также за авторизацию пользователей и взаимодействие с ними. Он обеспечивает интуитивно понятный интерфейс для пользователей, позволяя им взаимодействовать с элементами игры, такими как кнопки и меню.
2. **Логический модуль**: Модуль, который контролирует выполнение правил игры, проверку условий победы и учет ходов игроков. Этот модуль следит за состоянием игры, проверяет, кто победил или зафиксировалась ли ничья, а также управляет ходами игроков и очередностью их действий.
3. **Модуль авторизации пользователей**: Данный модуль включает функции регистрации, входа и выхода из аккаунта. Он обеспечивает безопасность и управление пользователями, сохраняя данные пользователей в безопасном виде и поддерживая процесс авторизации и регистрации в приложении.
4. **Модуль сохранения и загрузки состояния игры**: Этот модуль отвечает за сохранение текущего состояния игры, включая позицию игроков и количество ходов, а также за загрузку сохраненных данных для продолжения игры. Он также сохраняет и загружает данные пользователей, такие как статистика и настройки игры.

Модульная структура приложения позволяет эффективно организовать работу различных компонентов и обеспечивает гибкость для расширения функционала в будущем.

**2.2 Особенности реализации приложения**

В программе используются различные структуры данных и алгоритмы для обеспечения функциональности и стабильности работы.

1. **Структуры данных**:
   * Для хранения состояния игрового поля используется двумерный массив. Каждая ячейка массива представляет собой клетку игрового поля, которая может быть пустой, занята крестиком или ноликом.
   * Для реализации авторизации пользователей используется структура данных, которая хранит учетные записи пользователей в виде JSON-файла. Это позволяет сохранять данные о пользователях, такие как имя, хешированный пароль и историю побед.
2. **Обработка ошибок**:
   * Программа включает обработку некорректного ввода, например, попытки сделать ход на уже занятую клетку или ввод неверных данных при регистрации/входе в аккаунт. Это обеспечивает стабильность работы и предотвращает возможные сбои при взаимодействии с пользователем.
3. **Алгоритм игры против компьютера**:
   * Для реализации игры против компьютера используется алгоритм минимакса с эвристической оценкой позиций. Алгоритм минимакса позволяет компьютеру выбирать оптимальные ходы, оценивая возможные позиции на игровом поле и принимая наилучшее решение для выигрыша или защиты от проигрыша. Эвристическая оценка позволяет ускорить процесс выбора ходов, избегая полного перебора возможных вариантов.
4. **Модули сохранения и загрузки состояния**:
   * Для сохранения состояния игры используется формат JSON, что позволяет эффективно сохранять и загружать информацию о текущем состоянии игры, а также данные пользователей. Это позволяет продолжить игру с того места, где она была завершена, и сохранить статистику игр.

Модульная архитектура и использование эффективных алгоритмов и структур данных делают приложение стабильным, удобным в использовании и гибким для расширения.

**3. Обращение к программе**

**3.1 Запуск программы**

Для запуска приложения необходимо выполнить следующие шаги:

Убедитесь, что на компьютере установлен Python версии 3.8 или выше.

Запустите основной файл программы (main.py) через командную строку или IDE.

При открывании программы сначала появляется окно авторизации: пользователю необходимо войти в систему или зарегистрироваться, чтобы продолжить. После успешной авторизации откроется главное меню, где можно выбрать режим игры: против другого игрока или против компьютера, а также задать параметры игры, такие как размер поля.

**3.2 Правила игры**

Игроки делают ходы поочерёдно.

Для победы требуется расположить 5 своих фигур в ряд (по горизонтали, вертикали или диагонали).

Если игровое поле заполнено, а победитель не определён, фиксируется ничья.

**3.3 Управление**

Программа поддерживает два режима игры:

1. Против игрока: Два пользователя поочерёдно делают ходы на одном устройстве.
2. Против компьютера: Игрок соревнуется с компьютером, который принимает решения на основе алгоритма минимакса.

Для размещения фигуры достаточно кликнуть на свободную клетку поля.

**4. Сообщения**

Во время игры приложение выводит различные сообщения и уведомления:

**Сообщения о текущем ходе игры:** программа информирует игроков о том, чей сейчас ход.

**Уведомления о победе одного из участников или ничьей:** после завершения партии выводится сообщение о результате игры.

**Предупреждения об ошибочных действиях:** если игрок пытается сделать недопустимый ход, например, поставить фигуру на уже занятую клетку.

**Состояние игры после каждого хода:** после каждого действия игрока обновляется информация о текущем положении фигур на игровом поле.

# **Заключение**

В ходе разработки приложения «Крестики-Нолики до 5 в ряд» была создана полноценная компьютерная версия игры, которая соответствует всем требованиям, изложенным в техническом задании. Программа предоставляет пользователю возможность играть как против другого человека, так и против компьютера, используя алгоритм минимакса для выбора оптимальных ходов.

Все ключевые функции, такие как авторизация пользователей, выбор режима игры, определение победителя, а также сохранение и загрузка состояния игры, были реализованы в строгом соответствии с требованиями ТЗ. Приложение обеспечивает стабильную работу, обработку ошибок и защиту данных пользователей.

Тестирование показало, что приложение функционирует корректно во всех предусмотренных сценариях, включая работу алгоритма игры против компьютера и обработку всех возможных ситуаций в игре. Реализованные механизмы защиты, такие как хеширование паролей, обеспечивают безопасность данных.

Таким образом, разработанное приложение полностью соответствует целям и задачам, поставленным в техническом задании, и представляет собой качественный инструмент для игры «Крестики-Нолики до 5 в ряд».

# **Список использованных источников**

1. Difficulty of Various Games for Computers [Электронный ресурс]. – Режим доступа: URL: <https://blog.carlmjohnson.net/post/xkcd-game-ais-onceagain-a-webcomic/> (дата обращения : 29.10.2024)
2. Российский Институт Стандартизации // ГОСТ 19 ЕСПД (ГОСТ 19) Единая система программной документации [Электронный ресурс]. – Режим доступа: URL: <https://www.standards.ru/collection.aspx?control=40&id=868075&catalogid=temat-sbor/> (дата обращения : 29.10.2024)
3. Зараменских, Е. П. Управление жизненным циклом информационных систем: учебник и практикум для вузов / Е. П. Зараменских. – 2-е изд. – Москва: Издательство «Юрайт», 2023. – 497 с. – Текст: электронный // Образовательная платформа «Юрайт» [сайт]. [Электронный ресурс]. – Режим доступа: URL: <https://urait.ru/bcode/511960> (дата обращения : 29.10.2024)
4. Стандарты и шаблоны для ТЗ на разработку ПО [Электронный ресурс]. – Режим доступа: URL: https://habr.com/ru/articles/328822/ (дата обращения: 29.10.2024)
5. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению [Электронный ресурс]. – Режим доступа: URL: <https://www.prj-exp.ru/gost/gost_19-201-78.php> (дата обращения: 29.10.2024)
6. Крестики-Нолики (игра) [Электронный ресурс]. – Режим доступа: URL:
7. https://ru.wikipedia.org/wiki/Крестики-нолики (дата обращения: 29.10.2024)
8. Вирт Н. Алгоритмы + структуры данных = программы: Пер. с англ. / Н. Вирт. – Москва : Мир, 1985. – 406 с.
9. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р.Мартин. – Санкт-Петербург : Питер, 2022. – 352 с.
10. Архитектура программного обеспечения [Электронный ресурс]. – Режим доступа: URL: https://dic.academic.ru/dic.nsf/ruwiki/146913 (дата обращения: 29.10.2024)
11. Josh Smith / Creating Tic-tac-toe in Swift: User interface [Электронный ресурс]. – Режим доступа: URL: <https://ijoshsmith.com/2015/12/22/creatingtic-tac-toe-in-swift-user-interface/> (дата обращения : 29.10.2024)
12. Документация библиотеки tkinter [Электронный ресурс]. – Режим доступа: URL: <https://tkdocs.com/index.html> (дата обращения : 29.10.2024)
13. Классические технологии проектирования программ [Электронный ресурс]. Режим доступа: URL: [https://studfile.net/preview/1444530/page:14/#31](https://studfile.net/preview/1444530/page:14/" \l "31) (дата обращения: 29.10.2024)
14. Доусон, М. Программируем на Pyton / М.Доусон. – Санкт-Петербург: Питер, 2021. – 416 с.
15. Большой учебник по тестированию [Электронный ресурс]. – Доступ: URL: [https://qarocks.ru/big-software-testing-textbook/#fundamentals](https://qarocks.ru/big-software-testing-textbook/" \l "fundamentals) (дата обращения : 29.10.2024)
16. Назина, О.Е. Что такое тестирование. Курс молодого бойца / О.Е. Назина. Санкт-Петербург : БХВ-Петербург, 2022. 592 с.: ил.
17. Хорников, В. Принципы юнит-тестирования / В.Хорников. –СанктПетербург : Питер, 2021. – 320 с.: ил. – (Серия «Для профессионалов»).
18. Документация библиотеки Unittest [Электронный ресурс]. – Режим доступа: URL: <https://docs.python.org/3/library/unittest.html> (дата обращения: 29.10.2024)

# **Текст программы**

Листинг 1. «main.py»

import hashlib

import json

import os

import time

import tkinter as tk

from enum import Enum

from tkinter import ttk, messagebox

class ModeState(Enum):

computer = "Против компьютера"

player = "Против игрока"

class TicTacToe:

def \_\_init\_\_(self, size=10, mode: ModeState = ModeState.computer):

self.size = size

self.board = [[0] \* size for \_ in range(size)]

self.current\_player = 1

self.winning\_length = 5

self.directions = [(0, 1), (1, 0), (1, 1), (1, -1)]

self.mode = mode

self.game\_over = False

self.moves = 0

def is\_moves\_left(self) -> bool:

"""Проверка, на оставшиеся ходы"""

for row in self.board:

if row.count(0):

return False

return True

def make\_move(self, row, col) -> bool:

"""Сделать ход."""

if self.game\_over:

return False

if self.board[row][col] == 0:

self.board[row][col] = self.current\_player

self.current\_player = -self.current\_player

self.moves += 1

Продолжение листинг 1.

if self.mode == ModeState.computer and self.current\_player == -1:

self.make\_computer\_move()

if self.check\_winner(1) or self.check\_winner(-1):

self.game\_over = True

return True

return False

def make\_computer\_move(self):

"""Сделать ход компьютера."""

row, col = self.get\_best\_move()

self.make\_move(row, col)

def evaluate\_shape(self, consecutive, open\_ends, current\_player):

"""Оценивает комбинацию на основе последовательных фигур и открытых концов."""

if open\_ends == 0 and consecutive < 5:

# Если нет открытых концов и менее 5 в ряд - комбинация бесполезна

return 0

scores = {

# 4 фигуры в ряд

4: {

1: 50 if not current\_player else 100000000, # Один открытый конец

2: 500000 if not current\_player else 100000000 # Два открытых конца

},

# 3 фигуры в ряд

3: {

1: 5 if not current\_player else 7, # Один открытый конец

2: 50 if not current\_player else 10000 # Два открытых конца

},

# 2 фигуры в ряд

2: {

1: 2, # Один открытый конец

2: 5 # Два открытых конца

},

# 1 фигура в ряд

1: {

1: 0.5, # Один открытый конец

2: 1 # Два открытых конца

}

}

Продолжение листинг 1.

if consecutive >= 5:

# Выигрышная последовательность

return 200000000

return scores.get(consecutive, {}).get(open\_ends, 0)

def analyze\_direction(self, row, col, direction, player):

"""Анализирует линию в заданном направлении для подсчета очков."""

consecutive = 0 # Счетчик последовательных фигур

open\_ends = 0 # Счетчик открытых концов

score = 0 # Оценка линии

# Проверка открытого конца сзади

prev\_row = row - direction[0]

prev\_col = col - direction[1]

if (0 <= prev\_row < self.size and

0 <= prev\_col < self.size and

self.board[prev\_row][prev\_col] == 0):

open\_ends += 1

# Подсчет последовательных фигур

while (0 <= row < self.size and

0 <= col < self.size and

self.board[row][col] == player):

consecutive += 1

row += direction[0]

col += direction[1]

# Проверка открытого конца спереди

if (0 <= row < self.size and 0 <= col < self.size and self.board[row][col] == 0):

open\_ends += 1

if consecutive > 0:

# Если есть последовательные фигуры, тогда оцениваем комбинацию на основе последовательных фигур и открытых концов

score = self.evaluate\_shape(consecutive, open\_ends, player == self.current\_player)

return score

def evaluate\_position(self):

"""Оценка всех позиций."""

Продолжение листинг 1.

score = 0

checked\_lines = set()

for row in range(self.size):

for col in range(self.size):

if self.board[row][col] != 0:

player = self.board[row][col]

for direction in self.directions:

line\_key = (row, col, direction[0], direction[1])

if line\_key not in checked\_lines:

checked\_lines.add(line\_key)

score += self.analyze\_direction(row, col, direction, player) \* player

return score

def get\_valid\_moves(self):

"""Получает список возможных ходов с приоритизацией и исключением бесполезных ходов"""

moves = []

for row in range(self.size):

for col in range(self.size):

if self.board[row][col] == 0:

priority = 0

# Проверяем соседние клетки в радиусе 2

for dr in range(-2, 3):

for dc in range(-2, 3):

r, c = row + dr, col + dc

if 0 <= r < self.size and 0 <= c < self.size:

if self.board[r][c] != 0:

# Ближайшие соседи имеют больший вес

priority += 3 if abs(dr) <= 1 and abs(dc) <= 1 else 1

if priority > 0:

moves.append((priority, row, col))

moves.sort(reverse=True)

return [(row, col) for \_, row, col in moves] if moves else [(self.size // 2, self.size // 2)]

def has\_neighbor(self, row, col):

"""Проверка, есть ли рядом фигура."""

for dr in [-1, 0, 1]:

Продолжение листинг 1.

for dc in [-1, 0, 1]:

if dr == 0 and dc == 0:

continue

r, c = row + dr, col + dc

if (0 <= r < self.size and

0 <= c < self.size and

self.board[r][c] != 0):

return True

return False

def minimax(self, depth, alpha=float('-inf'), beta=float('inf'), maximizing\_player=False):

"""Минимакс алгоритм для выбора наилучшего хода."""

if depth == 0:

return self.evaluate\_position(), None

valid\_moves = self.get\_valid\_moves()

if not valid\_moves:

return 0, None

best\_move = None

if maximizing\_player:

max\_eval = float('-inf')

for move in valid\_moves:

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

eval\_score, \_ = self.minimax(depth - 1, alpha, beta, False)

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

if eval\_score > max\_eval:

max\_eval = eval\_score

best\_move = move

alpha = max(alpha, eval\_score)

if beta <= alpha:

break

return max\_eval, best\_move

else:

min\_eval = float('inf')

for move in valid\_moves:

self.board[move[0]][move[1]] = -1

eval\_score, \_ = self.minimax(depth - 1, alpha, beta, True)

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

Продолжение листинг 1.

if eval\_score < min\_eval:

min\_eval = eval\_score

best\_move = move

beta = min(beta, eval\_score)

if beta <= alpha:

break

return min\_eval, best\_move

def get\_best\_move(self):

"""Вычисление лучшего хода для компьютера."""

\_, move = self.minimax(2)

return move

def check\_winner(self, player):

"""Проверка выиграл ли игрок."""

for row in range(self.size):

for col in range(self.size):

if self.board[row][col] == player:

for direction in self.directions:

count = 1

r, c = row + direction[0], col + direction[1]

while (0 <= r < self.size and

0 <= c < self.size and

self.board[r][c] == player):

count += 1

if count == self.winning\_length:

return True

r += direction[0]

c += direction[1]

return False

class AuthService:

def \_\_init\_\_(self):

self.users\_file = "users.json"

self.users = {}

self.load\_users()

def login(self, username, password):

if username not in self.users:

Продолжение листинг 1.

return False, "Неверное имя пользователя"

if AuthService.verify\_password(password, self.users[username]):

return True, "Успешный вход"

return False, "Неверный пароль"

def register(self, username, password, confirm\_password):

if username in self.users:

return False, "Такое имя пользователя уже зарегистрировано"

if password != confirm\_password:

return False, "Пароли не совпадают"

self.users[username] = AuthService.hash\_password(password)

self.save\_users()

return True, "Успешная регистрация"

@staticmethod

def hash\_password(password):

return hashlib.sha256(password.encode()).hexdigest()

@staticmethod

def verify\_password(password, hashed\_password):

return AuthService.hash\_password(password) == hashed\_password

def save\_users(self):

with open(self.users\_file, "w") as f:

json.dump(self.users, f)

def load\_users(self):

if os.path.exists(self.users\_file):

with open(self.users\_file, "r") as f:

self.users = json.load(f)

class AuthWindow(tk.Toplevel):

def \_\_init\_\_(self, root):

super().\_\_init\_\_(root)

self.root = root

Продолжение листинг 1.

self.resizable(False, False)

self.grab\_set()

self.focus\_set()

self.transient(root)

self.auth\_service = AuthService()

self.main\_frame = tk.Frame(self)

self.main\_frame.pack(expand=True, fill='both', padx=20, pady=20)

self.fields = {}

def create\_field(self, label, field):

frame = tk.Frame(self.main\_frame)

frame.pack(fill=tk.X, pady=10)

label = tk.Label(frame, text=label)

label.pack(anchor=tk.W)

entry = tk.Entry(frame)

entry.pack(fill=tk.X, expand=True)

error\_label = tk.Label(frame, text="", fg="red")

error\_label.pack(anchor=tk.W)

self.fields[field] = (entry, error\_label)

def get\_field(self, field):

return self.fields[field][0].get()

def hide\_input(self, field):

self.fields[field][0].config(show="\*")

def show\_input(self, field):

self.fields[field][0].config(show="")

def show\_error(self, field, message):

self.fields[field][1].config(text=message)

def clear\_error(self, field):

self.fields[field][1].config(text="")

Продолжение листинг 1.

class LoginWindow(AuthWindow):

def \_\_init\_\_(self, root, on\_success=None):

super().\_\_init\_\_(root)

self.title("Авторизация")

self.geometry("300x300")

self.create\_field("Имя пользователя:", "username")

self.create\_field("Пароль:", "password")

self.hide\_input("password")

self.on\_success = on\_success

show\_password\_var = tk.BooleanVar()

show\_password\_checkbox = tk.Checkbutton(

self.main\_frame,

text="Показать пароль",

variable=show\_password\_var,

command=lambda: self.toggle\_password\_visibility(show\_password\_var.get())

)

show\_password\_checkbox.pack(anchor=tk.W, pady=10)

buttons\_frame = tk.Frame(self.main\_frame)

buttons\_frame.pack(pady=20)

self.login\_button = tk.Button(buttons\_frame, text="Войти", command=self.login)

self.login\_button.pack(side=tk.LEFT, padx=5)

self.register\_button = tk.Button(buttons\_frame, text="Регистрация",

command=self.show\_register)

self.register\_button.pack(side=tk.LEFT, padx=5)

def toggle\_password\_visibility(self, show\_password):

if show\_password:

self.show\_input("password")

else:

self.hide\_input("password")

def login(self):

username = self.get\_field("username")

password = self.get\_field("password")

Продолжение листинг 1.

self.clear\_error("username")

self.clear\_error("password")

if not username:

self.show\_error("username", "Введите имя пользователя")

return

if not password:

self.show\_error("password", "Введите пароль")

return

success, message = self.auth\_service.login(username, password)

if success:

if self.on\_success:

self.on\_success(username)

self.destroy()

else:

if "пользователя" in message:

self.show\_error("username", message)

else:

self.show\_error("password", message)

def show\_register(self):

self.destroy()

RegisterWindow(self.root, self.on\_success)

class RegisterWindow(AuthWindow):

def \_\_init\_\_(self, root, on\_success=None):

super().\_\_init\_\_(root)

self.title("Регистрация")

self.geometry("300x400")

self.on\_success = on\_success

self.create\_field("Имя пользователя:", "username")

self.create\_field("Пароль:", "password")

self.create\_field("Подтвердите пароль:", "confirm\_password")

self.hide\_input("password")

self.hide\_input("confirm\_password")

Продолжение листинг 1.

show\_password\_var = tk.BooleanVar()

show\_password\_checkbox = tk.Checkbutton(

self.main\_frame,

text="Показать пароль",

variable=show\_password\_var,

command=lambda: self.toggle\_password\_visibility(show\_password\_var.get())

)

show\_password\_checkbox.pack(anchor=tk.W, pady=10)

buttons\_frame = tk.Frame(self.main\_frame)

buttons\_frame.pack(pady=20)

self.register\_button = tk.Button(buttons\_frame, text="Зарегистрироваться", command=self.register\_user)

self.register\_button.pack(padx=5)

self.back\_button = tk.Button(buttons\_frame, text="Назад", command=self.back\_to\_login)

self.back\_button.pack(padx=5)

def toggle\_password\_visibility(self, show\_password):

if show\_password:

self.show\_input("password")

self.show\_input("confirm\_password")

else:

self.hide\_input("password")

self.hide\_input("confirm\_password")

def register\_user(self):

username = self.get\_field("username")

password = self.get\_field("password")

confirm\_password = self.get\_field("confirm\_password")

for field in self.fields:

self.clear\_error(field)

if not username:

self.show\_error("username", "Введите имя пользователя")

return

if len(username) < 3:

Продолжение листинг 1.

self.show\_error("username", "Имя пользователя должно быть не менее 3 символов")

return

if not password:

self.show\_error("password", "Введите пароль")

return

if len(password) < 6:

self.show\_error("password", "Пароль должен быть не менее 6 символов")

return

if not confirm\_password:

self.show\_error("confirm\_password", "Подтвердите пароль")

return

success, message = self.auth\_service.register(username, password, confirm\_password)

if success:

messagebox.showinfo("Успех", message)

self.back\_to\_login()

else:

if "пользователя" in message:

self.show\_error("username", message)

elif "Пароли" in message:

self.show\_error("confirm\_password", message)

else:

self.show\_error("password", message)

def back\_to\_login(self):

self.destroy()

LoginWindow(self.root, self.on\_success)

class TicTacToeApp:

def \_\_init\_\_(self, root):

self.root = root

self.root.title("Крестики-Нолики до 5 в ряд")

self.game = TicTacToe()

self.cell\_size = 65

self.game\_mode = ModeState.computer

Продолжение листинг 1.

self.start\_time = None

self.timer\_id = None

self.sidebar\_frame = tk.Frame(self.root, width=200)

self.sidebar\_frame.pack(side=tk.LEFT, fill=tk.Y)

self.sidebar\_frame.pack\_propagate(False)

self.main\_frame = tk.Frame(self.root, padx=10, pady=10)

self.main\_frame.pack(expand=True, fill=tk.BOTH)

settings\_frame = tk.LabelFrame(self.sidebar\_frame, text="Настройки игры", padx=10, pady=5)

settings\_frame.pack(fill=tk.X, padx=5, pady=5)

self.mode\_label = tk.Label(settings\_frame, text="Режим игры:")

self.mode\_label.pack(anchor=tk.W, pady=(5, 0))

self.mode\_var = tk.StringVar(value=ModeState.computer.value)

self.mode\_menu = ttk.OptionMenu(

settings\_frame,

self.mode\_var,

ModeState.computer.value,

\*[mode.value for mode in ModeState],

command=self.change\_mode

)

self.mode\_menu.pack(fill=tk.X, pady=(0, 10))

control\_frame = tk.LabelFrame(self.sidebar\_frame, text="Управление", padx=10, pady=5)

control\_frame.pack(fill=tk.X, padx=5, pady=5)

self.reset\_button = tk.Button(control\_frame, text="Новая игра", command=self.reset\_game)

self.reset\_button.pack(fill=tk.X, pady=5)

stats\_frame = tk.LabelFrame(self.sidebar\_frame, text="Статистика", padx=10, pady=5)

stats\_frame.pack(fill=tk.X, padx=5, pady=5)

self.status\_var = tk.StringVar(value="Ход: X")

Продолжение листинг 1.

self.status\_label = tk.Label(stats\_frame, textvariable=self.status\_var)

self.status\_label.pack(anchor=tk.W, pady=5)

self.moves\_var = tk.StringVar(value="Ходов: 0")

self.moves\_label = tk.Label(stats\_frame, textvariable=self.moves\_var)

self.moves\_label.pack(anchor=tk.W, pady=5)

self.time\_var = tk.StringVar(value="Время: 00:00")

self.time\_label = tk.Label(stats\_frame, textvariable=self.time\_var)

self.time\_label.pack(anchor=tk.W, pady=5)

player\_frame = tk.LabelFrame(self.sidebar\_frame, text="Игрок", padx=10, pady=5)

player\_frame.pack(fill=tk.X, padx=5, pady=5)

self.player\_name\_var = tk.StringVar(value="Гость")

self.player\_label = tk.Label(player\_frame, textvariable=self.player\_name\_var)

self.player\_label.pack(anchor=tk.W, pady=5)

self.logout\_button = tk.Button(player\_frame, text="Выйти", command=self.logout)

self.logout\_button.pack(fill=tk.X, pady=5)

canvas\_size = self.cell\_size \* self.game.size

self.canvas = tk.Canvas(

self.main\_frame,

width=canvas\_size,

height=canvas\_size,

bg="white"

)

self.canvas.pack()

self.canvas.bind("<Button-1>", self.on\_click)

self.draw\_board()

LoginWindow(self.root, self.on\_login)

def draw\_board(self):

"""Отрисовка игрового поля."""

self.canvas.delete("all")

for i in range(self.game.size + 1):

Продолжение листинг 1.

self.canvas.create\_line(

i \* self.cell\_size, 0,

i \* self.cell\_size, self.game.size \* self.cell\_size,

fill="gray"

)

self.canvas.create\_line(

0, i \* self.cell\_size,

self.game.size \* self.cell\_size, i \* self.cell\_size,

fill="gray"

)

for row in range(self.game.size):

for col in range(self.game.size):

if self.game.board[row][col] == 1:

self.draw\_player(row, col)

elif self.game.board[row][col] == -1:

self.draw\_opponent(row, col)

def draw\_player(self, row, col):

"""Отрисовка фигуры игрока."""

x = col \* self.cell\_size

y = row \* self.cell\_size

padding = self.cell\_size \* 0.2

color = "blue"

thickness = 2

self.canvas.create\_line(

x + padding, y + padding,

x + self.cell\_size - padding, y + self.cell\_size - padding,

width=thickness,

fill=color

)

self.canvas.create\_line(

x + self.cell\_size - padding, y + padding,

x + padding, y + self.cell\_size - padding,

width=thickness,

fill=color

)

def draw\_opponent(self, row, col):

"""Отрисовка фигуры противника."""

Продолжение листинг 1.

x = col \* self.cell\_size

y = row \* self.cell\_size

padding = self.cell\_size \* 0.2

color = "red"

thickness = 2

self.canvas.create\_oval(

x + padding, y + padding,

x + self.cell\_size - padding, y + self.cell\_size - padding,

width=thickness,

outline=color

)

def on\_click(self, event):

"""Обработчик нажатия на игровое поле."""

row, col = event.y // self.cell\_size, event.x // self.cell\_size

if self.game.make\_move(row, col):

self.moves\_var.set(f"Ходов: {self.game.moves}")

self.draw\_board()

if self.game.check\_winner(1):

# x, y = self.canvas.winfo\_width() / 2, self.canvas.winfo\_height() / 2

# self.canvas.create\_text(x, y, text="Крестики победили!", font="Arial 32")

messagebox.showinfo("Победа!", "Крестики победили!")

elif self.game.check\_winner(-1):

# x, y = self.canvas.winfo\_width() / 2, self.canvas.winfo\_height() / 2

# self.canvas.create\_text(x, y, text="Нолики победили!", font="Arial 32")

messagebox.showinfo("Поражение!", "Нолики победили!")

elif self.game.is\_moves\_left():

messagebox.showinfo("Ничья!", "Ничья!")

def change\_mode(self, event):

"""Обработчик изменения режима игры."""

self.game\_mode = ModeState(self.mode\_var.get())

self.reset\_game()

def change\_size(self):

"""Обработчик изменения размера игрового поля."""

self.game\_size = self.size\_var.get()

self.reset\_game()

def reset\_game(self):

Продолжение листинг 1.

"""Сброс игры."""

self.game = TicTacToe(mode=self.game\_mode)

self.draw\_board()

self.moves\_var.set(f"Ходов: {self.game.moves}")

self.time\_var.set(f"Время: 00:00")

self.status\_var.set("X - ход")

self.start\_time = time.time()

self.update\_timer()

def update\_timer(self):

"""Обновление таймера."""

if self.start\_time is None:

self.start\_time = time.time()

elapsed = int(time.time() - self.start\_time)

minutes = elapsed // 60

seconds = elapsed % 60

self.time\_var.set(f"Время: {minutes:02d}:{seconds:02d}")

if not self.game.game\_over:

self.timer\_id = self.root.after(1000, self.update\_timer)

def update\_status(self):

"""Обновление состояния игры."""

if self.game.game\_over:

winner = "X" if self.game.current\_player == -1 else "O"

self.status\_var.set(f"{winner} победили!")

else:

current = "X" if self.game.current\_player == 1 else "O"

self.status\_var.set(f"{current} - ход")

def logout(self):

self.player\_name\_var.set("Гость")

LoginWindow(self.root, self.on\_login)

def on\_login(self, username):

self.player\_name\_var.set(username)

def run(self):

self.root.mainloop()

Продолжение листинг 1.

if \_\_name\_\_ == "\_\_main\_\_":

root = tk.Tk()

app = TicTacToeApp(root)

app.run()

Листинг 2. «test\_TicTacToe.py»

import os

import unittest

from main import TicTacToe, ModeState, AuthService

class TestAuthService(unittest.TestCase):

def setUp(self):

self.auth\_service = AuthService()

self.test\_users\_file = "test\_users.json"

self.auth\_service.users\_file = self.test\_users\_file

def tearDown(self):

# Удаляем тестовый файл после тестов

if os.path.exists(self.test\_users\_file):

os.remove(self.test\_users\_file)

def test\_register(self):

# Тест успешной регистрации

success, message = self.auth\_service.register("testuser", "password123", "password123")

self.assertTrue(success)

self.assertEqual(message, "Успешная регистрация")

# Тест регистрации существующего пользователя

success, message = self.auth\_service.register("testuser", "password123", "password123")

self.assertFalse(success)

# Тест регистрации с несовпадающими паролями

Продолжение листинг 2.

success, message = self.auth\_service.register("newuser", "pass1", "pass2")

self.assertFalse(success)

def test\_login(self):

# Регистрируем тестового пользователя

self.auth\_service.register("testuser", "password123", "password123")

# Тест успешного входа

success, message = self.auth\_service.login("testuser", "password123")

self.assertTrue(success)

# Тест входа с неверным паролем

success, message = self.auth\_service.login("testuser", "wrongpass")

self.assertFalse(success)

# Тест входа несуществующего пользователя

success, message = self.auth\_service.login("nonexistent", "password123")

self.assertFalse(success)

def test\_password\_hashing(self):

# Тест хеширования пароля

password = "testpass123"

hashed = AuthService.hash\_password(password)

self.assertTrue(AuthService.verify\_password(password, hashed))

self.assertFalse(AuthService.verify\_password("wrongpass", hashed))

def test\_save\_load\_users(self):

# Тест сохранения и загрузки пользователей

self.auth\_service.register("testuser", "password123", "password123")

self.auth\_service.save\_users()

# Создаем новый экземпляр сервиса для загрузки данных

new\_service = AuthService()

new\_service.users\_file = self.test\_users\_file

new\_service.load\_users()

self.assertIn("testuser", new\_service.users)

class TestTicTacToe(unittest.TestCase):

Продолжение листинг 2.

def setUp(self):

# Создаем две игры с разными режимами для тестирования

self.game\_vs\_computer = TicTacToe(mode=ModeState.computer)

self.game\_vs\_player = TicTacToe(mode=ModeState.player)

def test\_initialization(self):

"""Тест инициализации игры"""

# Проверка режима против компьютера

self.assertEqual(self.game\_vs\_computer.size, 10)

self.assertEqual(self.game\_vs\_computer.current\_player, 1)

self.assertEqual(self.game\_vs\_computer.winning\_length, 5)

self.assertEqual(self.game\_vs\_computer.mode, ModeState.computer)

self.assertFalse(self.game\_vs\_computer.game\_over)

self.assertEqual(self.game\_vs\_computer.moves, 0)

# Проверка режима против игрока

self.assertEqual(self.game\_vs\_player.size, 10)

self.assertEqual(self.game\_vs\_player.current\_player, 1)

self.assertEqual(self.game\_vs\_player.mode, ModeState.player)

self.assertFalse(self.game\_vs\_player.game\_over)

def test\_make\_move\_vs\_computer(self):

"""Тест ходов в режиме против компьютера"""

# Ход игрока

self.assertTrue(self.game\_vs\_computer.make\_move(0, 0))

self.assertEqual(self.game\_vs\_computer.board[0][0], 1)

# Проверка, что после хода игрока компьютер сделал свой ход

computer\_moved = False

for row in self.game\_vs\_computer.board:

if -1 in row:

computer\_moved = True

break

self.assertTrue(computer\_moved)

# Попытка хода в занятую клетку

self.assertFalse(self.game\_vs\_computer.make\_move(0, 0))

def test\_make\_move\_vs\_player(self):

"""Тест ходов в режиме против игрока"""

Продолжение листинг 2.

# Ход первого игрока

self.assertTrue(self.game\_vs\_player.make\_move(0, 0))

self.assertEqual(self.game\_vs\_player.board[0][0], 1)

self.assertEqual(self.game\_vs\_player.current\_player, -1)

# Ход второго игрока

self.assertTrue(self.game\_vs\_player.make\_move(0, 1))

self.assertEqual(self.game\_vs\_player.board[0][1], -1)

self.assertEqual(self.game\_vs\_player.current\_player, 1)

def test\_check\_winner\_horizontal(self):

"""Тест проверки победителя по горизонтали"""

# Создаем выигрышную комбинацию по горизонтали

for i in range(5):

self.game\_vs\_player.board[0][i] = 1

self.assertTrue(self.game\_vs\_player.check\_winner(1))

def test\_check\_winner\_vertical(self):

"""Тест проверки победителя по вертикали"""

# Создаем выигрышную комбинацию по вертикали

for i in range(5):

self.game\_vs\_player.board[i][0] = 1

self.assertTrue(self.game\_vs\_player.check\_winner(1))

def test\_check\_winner\_diagonal(self):

"""Тест проверки победителя по диагонали"""

# Создаем выигрышную комбинацию по диагонали

for i in range(5):

self.game\_vs\_player.board[i][i] = 1

self.assertTrue(self.game\_vs\_player.check\_winner(1))

def test\_check\_winner\_reverse\_diagonal(self):

"""Тест проверки победителя по обратной диагонали"""

# Создаем выигрышную комбинацию по обратной диагонали

for i in range(5):

self.game\_vs\_player.board[i][4 - i] = 1

self.assertTrue(self.game\_vs\_player.check\_winner(1))

def test\_is\_moves\_left(self):

"""Тест проверки оставшихся ходов"""

Продолжение листинг 2.

# Заполняем все поле

for i in range(self.game\_vs\_player.size):

for j in range(self.game\_vs\_player.size):

self.game\_vs\_player.board[i][j] = 1

self.assertTrue(self.game\_vs\_player.is\_moves\_left())

def test\_computer\_strategy(self):

"""Тест стратегии компьютера"""

# Создаем ситуацию, где компьютер должен заблокировать победу игрока

for i in range(4):

self.game\_vs\_computer.board[0][i] = 1

# Делаем ход и проверяем, что компьютер заблокировал победу

self.game\_vs\_computer.current\_player = -1

self.game\_vs\_computer.make\_computer\_move()

blocked = False

if self.game\_vs\_computer.board[0][4] == -1:

blocked = True

self.assertTrue(blocked)

def test\_game\_over\_state(self):

"""Тест состояния окончания игры"""

# Создаем победную комбинацию

for i in range(5):

self.game\_vs\_player.board[0][i] = 1

# Делаем ход и проверяем, что игра завершилась

self.game\_vs\_player.make\_move(1, 0)

self.assertTrue(self.game\_vs\_player.game\_over)

def test\_evaluate\_position(self):

"""Тест оценки позиции"""

# Проверяем начальную позицию

self.assertEqual(self.game\_vs\_computer.evaluate\_position(), 0)

# Создаем преимущество для игрока

for i in range(3):

self.game\_vs\_computer.board[0][i] = 1

self.assertNotEqual(self.game\_vs\_computer.evaluate\_position(), 0)

def test\_get\_valid\_moves(self):

Продолжение листинг 2.

"""Тест получения возможных ходов"""

# Проверяем количество возможных ходов в начале игры

moves = self.game\_vs\_computer.get\_valid\_moves()

self.assertEqual(len(moves), 1) # В начале игры предлагается ход в центр

# Делаем ход и проверяем, что появились новые возможные ходы

self.game\_vs\_computer.make\_move(5, 5)

moves = self.game\_vs\_computer.get\_valid\_moves()

self.assertTrue(len(moves) > 1)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

# **ПРИЛОЖЕНИЕ 1**

