**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования   
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**Самолетостроительный факультет

Кафедра «Информационные технологии и общенаучные дисциплины»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

На лабораторную работу

По дисциплине «Алгоритмы и структуры данных»

ТемаРазработка логических компьютерных игр с графическим интерфейсом в среде Питон

Листов 19

Руководитель разработки:  
кандидат технических наук, доцент кафедры  
Шишкин В.В.   
«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

Исполнитель:  
студент гр. АИСТбд-21  
Травкина К.А.  
«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

2024г.

**Содержание**

Техническое задание...............................................................................................3

Пояснительная записка ..........................................................................................8

Руководство программиста ..................................................................................14

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования   
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**Самолетостроительный факультет

Кафедра «Информационные технологии и общенаучные дисциплины»

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

На лабораторную работу

По дисциплине «Алгоритмы и структуры данных»

ТемаРазработка логических компьютерных игр с графическим интерфейсом в среде Питон

Листов 5

Исполнитель:  
студент гр. АИСТбд-21  
Травкина К.А.  
«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

2024г.

**Введение**

**Наименование приложения**:

"Крестики-нолики" — компьютерная и многопользовательская версия игры с использованием алгоритма Минимакс для принятия решений.

**Условное обозначение**:

Приложение не имеет конкретного условного обозначения, так как это учебный проект, реализованный для лабораторной работы.

**Краткий свод правил игры**:

* Игра состоит из 9 клеток, размещённых в квадрате 3x3.
* Игроки поочередно ставят свои символы ("X" или "O") в пустые клетки.
* Побеждает тот, кто первым соберёт три одинаковых символа по вертикали, горизонтали или диагонали.
* Если все клетки заполняются без победителя, объявляется ничья.

**Общая характеристика функциональных возможностей**:

Приложение позволяет:

1. Играть в "Крестики-нолики" против другого игрока.
2. Играть в "Крестики-нолики" против компьютера, использующего алгоритм Минимакс для принятия решений.
3. Рестартовать игру и завершать её в любой момент.
4. Визуализировать состояние игры с помощью графического интерфейса на основе Tkinter.

**1. Основания для разработки**

В качестве оснований для разработки используется учебный план направления 09.03.02 «Информационные системы и технологии» и распоряжение по факультету. Данный проект был разработан в рамках выполнения курсовой работы

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

**2.1. Функциональное назначение**

Программа реализует классическую игру "Крестики-нолики" с возможностью игры между пользователем и компьютером, использующим алгоритм Минимакс для принятия решений. Программа автоматизирует следующие процессы:

1. Отображение игрового поля.
2. Ведение логики игры (поочерёдные ходы игроков, проверка на победу или ничью).
3. Обработка взаимодействия с пользователем через графический интерфейс.

Группы пользователей:

1. Игроки, принимающие участие в игре (человек и компьютер).
2. Программисты, которые могут модифицировать код игры для улучшения функционала.

#### ****2.2. Требования к функциональным характеристикам****

##### **2.2.1. Требования к структуре приложения**

Программа должна быть организована с использованием модульной структуры. Основные компоненты программы:

1. **Графический интерфейс** (GUI) — для взаимодействия с пользователем.
2. **Логика игры** — включает проверку победителя, ходов игроков, минимакс-алгоритм для принятия решений компьютером.
3. **Основные функции**:

* Проверка победителя.
* Проверка ничьей.
* Переключение хода между игроком и компьютером.

##### **2.2.2. Требования к составу функций приложения**

Основные функции:

* check\_winner(): Проверка победителя.
* check\_draw(): Проверка ничьей.
* game\_over(): Завершение игры с выводом результата.
* start\_new\_game(): Перезапуск игры.
* player\_turn(): Логика хода игрока.
* computer\_turn(): Логика хода компьютера.
* minimax(): Алгоритм Минимакс для вычисления оптимального хода компьютера.

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

Входные данные:

* Ввод игроком ходов через графический интерфейс.
* Выбор режима игры (игра крестиками или ноликами).
* Выходные данные:
* Отображение состояния игрового поля.
* Сообщения о победе, ничьей или продолжении игры.
* Окно с возможностью перезапуска игры или завершения.

Пользовательский интерфейс:

* Игровое поле представляет собой сетку 3x3, где каждая клетка — это кнопка.
* Для каждого хода игрока или компьютера соответствующая кнопка блокируется и обновляется с новым символом ("X" или "O").

**2.3. Требования к надежности**

Программа должна быть работоспособной в течение всей игры и не должна аварийно завершаться. В случае ошибок в работе игры (например, при неверном вводе), программа должна информировать пользователя о возможной ошибке, но не завершать работу.

* В случае сбоя (например, при ошибке обработки ходов или интерфейса), приложение должно корректно восстановиться без потери данных.

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

Программа должна работать в средах с установленным Python 3.x и поддерживать Tkinter как графическую библиотеку. Для работы не требуется дополнительных библиотек.

* Операционная система: Windows, Linux, macOS.
* Библиотеки: Tkinter (для GUI), Random (для случайных чисел), любые библиотеки Python для математических операций.

**2.5. Требования к маркировке и упаковке**

Маркировка и упаковка не требуется, так как приложение является учебным проектом.

**2.6. Требования к транспортированию и хранению**

* **Условия транспортирования**: Требования к условиям транспортирования не предъявляются.
* **Условия хранения**: Обеспечение свободного доступа к проекту в репозитории до окончания срока учебы.
* **Сроки хранения**: Срок хранения — до окончания срока учебы.

### **3. Требования к программной документации**

Требования к программной документации определяются заданием на лабораторную работу.

### **4. Стадии и этапы разработки**

Этапы разработки определяются заданием на лабораторную работу.

### **5. Порядок контроля и приемки**

Порядок контроля и приемки определяется заданием на курсовую работу. Программа должна быть проверена на выполнение всех требований, корректную работу игры и соответствие стандартам программирования.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования   
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**Самолетостроительный факультет

Кафедра «Информационные технологии и общенаучные дисциплины»

**Лабораторная работа**

**По дисциплине «Алгоритмы и структуры**

**данных»**

**Тема** Разработка логических компьютерных игр с графическим интерфейсом в среде Питон

**Пояснительная записка**

Листов 6

Исполнитель:  
студент гр. АИСТбд-21  
Травкина К.А.  
«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

2024г.

**Введение**

**Наименование приложения**:

Приложение "Крестики-нолики" для двух игроков или игрока против компьютера. В игре используется графический интерфейс, основанный на библиотеке Tkinter, и алгоритм Минимакс для принятия решений компьютером.

**Условное обозначение приложения**:

Данное приложение является курсовым проектом, не имеет стандартного условного обозначения.

**Наименование реализованной игры**:

Игра "Крестики-нолики".

**Описание и обоснование выбранного подхода**:

В разработке приложения использовалась классическая модель игры "Крестики-нолики", где два игрока поочередно ставят свои символы (X или O) на поле 3x3, и выигрывает тот, кто первым выстроит три своих символа в линию. Игрок может выбрать игру за крестики или нолики. В игре с компьютером применяется алгоритм Минимакс для оценки наилучшего хода, а также блокировки возможных победных комбинаций противника.

Алгоритм Минимакс был выбран из-за своей эффективности в играх с полной информацией и его способности находить оптимальные ходы в условиях ограниченной размерности поля (3x3). Он используется для того, чтобы компьютер мог не только выигрывать, но и предотвращать выигрыши пользователя.

**Краткое описание реализованного приложения**:

Программа реализует классическую игру "Крестики-нолики" с графическим интерфейсом. Пользователь может играть как крестиками, так и ноликами. В игре реализована проверка победных состояний и ничьей. Приложение предоставляет возможность перезапуска игры и завершения сеанса.

### **1. Проектная часть**

#### ****1.1. Постановка задачи на разработку приложения****

Задача на разработку приложения определяется заданием на лабораторную работу и детализируется в разработанном техническом задании (Приложение 1).

#### ****1.2. Математические методы****

Программная логика игры не требует сложных математических методов, однако для реализации оптимальных ходов компьютером используется алгоритм Минимакс. Этот алгоритм основан на дереве решений, где каждая вершина дерева представляет собой возможное состояние игры. Компьютер анализирует все возможные ходы, начиная от текущего состояния, и выбирает тот, который максимально увеличивает его шансы на победу или минимизирует вероятность победы противника.

**Обоснование выбора**:

Алгоритм Минимакс был выбран, так как он является стандартным решением для игр с полным перебором возможных состояний. Он оценивает все возможные позиции и выбирает оптимальную, основываясь на идее минимизации потерь (для игрока) или максимизации выигрыша (для компьютера). В данном случае, из-за небольших размеров поля (3x3), алгоритм Минимакс будет работать эффективно и обеспечит достаточно хорошую производительность.

#### ****1.3. Архитектура и алгоритмы****

##### **1.3.1. Архитектура**

Архитектура приложения включает в себя несколько основных компонентов:

1. **Графический интерфейс** (GUI), реализованный с помощью библиотеки Tkinter. Он отвечает за отображение игрового поля и взаимодействие с пользователем.
2. **Логика игры** — включает в себя функциональность для обработки ходов игроков, проверки победных состояний и ничьей, а также работу с алгоритмом Минимакс для игры с компьютером.
3. **Основные функции**:

* Проверка победителя (check\_winner).
* Проверка на ничью (check\_draw).
* Логика ходов игроков (player\_turn, computer\_turn).
* Логика работы алгоритма Минимакс для вычисления оптимального хода компьютера (minimax).
* Функции для начала новой игры и завершения сеанса (start\_new\_game, game\_over).

Основные данные, с которыми работает приложение:

* **Игровое поле**: двумерный список размером 3x3, представляющий собой сетку клеток.
* **Кнопки**: каждая клетка на поле — это кнопка, которая обновляется при каждом ходе.

##### **1.3.2. Алгоритм проверки победителя и ничьей**

**Алгоритм проверки победителя (check\_winner)**:  
Алгоритм проверяет все возможные линии на поле (горизонтальные, вертикальные и диагональные) на предмет того, что все клетки содержат одинаковый символ ("X" или "O"). Если такая линия найдена, игра завершается победой игрока.

**Алгоритм проверки ничьей (check\_draw)**:

Алгоритм проверяет, что все клетки на поле заполнены, и, если победителя нет, объявляется ничья.

**Схема алгоритма check\_winner**:

1. Проверить все горизонтальные линии.
2. Проверить все вертикальные линии.
3. Проверить две диагонали.
4. Если все клетки одинаковы на какой-либо линии — победитель найден.
5. Если поле полностью заполнено, но победителя нет — ничья.

##### **1.3.3. Алгоритм Минимакс для компьютера**

**Алгоритм Минимакс (minimax)**:  
Этот алгоритм используется для нахождения оптимального хода для компьютера. Алгоритм выполняет рекурсивный обход дерева возможных состояний игры и оценивает каждую позицию. Алгоритм применяет принцип максимизации (для компьютера) и минимизации (для игрока), используя метод альфа-бета отсечения для повышения эффективности.

**Схема алгоритма Минимакс**:

1. Для каждого возможного хода рекурсивно оценивается состояние игры.
2. При глубине рекурсии 0 или при нахождении победного состояния возвращается оценка (1 для победы компьютера, -1 для победы игрока, 0 для ничьей).
3. Выбирается ход с максимальной или минимальной оценкой в зависимости от того, чья очередь ходить.
4. Применяется альфа-бета отсечение для уменьшения числа анализируемых состояний.

#### ****1.4. Тестирование****

##### **1.4.1. Описание отчета о тестировании**

Отчет о тестировании должен предоставить информацию о всех проведённых тестах, их результатах и рекомендациях для улучшения программы. Тестирование фокусируется на проверке корректности алгоритмов игры, стабильности работы интерфейса и логики игры.

##### **1.4.2. Цель тестирования**

Цель тестирования — убедиться, что программа работает без сбоев, корректно обрабатывает все типичные игровые ситуации (победа, ничья, смена игроков), а также что алгоритм Минимакс правильно принимает решения, учитывая все возможные ходы.

##### **1.4.3. Методика тестирования**

Методика тестирования включает в себя следующие этапы:

1. Тестирование графического интерфейса — проверка отображения и взаимодействия с кнопками.
2. Тестирование логики игры — проверка правильности работы функций для проверки победы и ничьей.
3. Тестирование алгоритма Минимакс — проверка правильности выбора хода компьютером.

##### **1.4.4. Проведенные тесты**

Тесты могут включать следующие сценарии:

* Проверка игры против компьютера.
* Проверка всех возможных ситуаций победы, ничьей и продолжения игры.
* Проверка правильности работы алгоритма Минимакс (выбор хода компьютером).

##### **1.4.5. Выводы**

В результате тестирования был подтвержден правильный выбор ходов компьютером, а также отсутствие ошибок в обработке игровых ситуаций. Рекомендации: можно улучшить графический интерфейс и добавить возможность настроек для игры (выбор уровня сложности).

**2. Источники, использованные при разработке**

1. ГОСТ 19.701-90 "Схемы алгоритмов, программ, данных и систем".
2. Шишкин В.В., Афонин Д.С. Разработка логических компьютерных игр с графическим интерфейсом в среде Питон. - УлГТУ, 2023

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования   
**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**Самолетостроительный факультет

Кафедра «Информационные технологии и общенаучные дисциплины»

**Лабораторная работа**

**По дисциплине «Алгоритмы и структуры**

**данных»**

**Тема** Разработка логических компьютерных игр с графическим интерфейсом в среде Питон

**Руководство программиста**

Листов 6

Исполнитель:  
студент гр. АИСТбд-21  
Травкина К.А.  
«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024г.

2024г.

### **1. Назначение и условия применения программы**

#### ****1.1. Назначение и функции, выполняемые приложением****

**Назначение приложения**:

Приложение предназначено для реализации игры "Крестики-нолики" с графическим интерфейсом для игры против компьютера. Игроки поочередно ставят свои символы ("X" или "O") на игровое поле размером 3x3, и выигрывает тот, кто первым выстроит три своих символа в ряд (горизонтально, вертикально или по диагонали).

**Свод правил игры**:

* Игроки по очереди ставят свои символы на поле.
* Символы могут быть только "X" или "O".
* Победа наступает, когда один из игроков ставит три своих символа в ряд (горизонтально, вертикально или по диагонали).
* Игра может закончиться ничьей, если все клетки заняты, но победитель не найден.

**Функциональные возможности**:

* Поддержка игры между двумя игроками на одном устройстве.
* Возможность игры с компьютером, который использует алгоритм Минимакс для принятия оптимальных решений.
* Отображение игрового поля и статуса игры в реальном времени.
* Подсчет победителя или ничьей.
* Перезапуск игры после завершения.

#### ****1.2. Условия, необходимые для использования приложения****

**Операционная система**:

* Приложение работает на любой операционной системе, поддерживающей Python 3.x. Это может быть Windows, macOS или Linux.

**Платформа и инструментальная среда**:

* Python версии 3.x.
* Среда разработки: любая IDE, поддерживающая Python (например, PyCharm, VS Code).

**Необходимые библиотеки**:

* **Tkinter**: стандартная библиотека для создания графического интерфейса.
* **math** (для вычислений в алгоритме Минимакс, если потребуется).
* Приложение не требует установки дополнительных библиотек или внешних зависимостей, кроме стандартной библиотеки Python.

### **2. Характеристики программы**

#### ****2.1. Характеристики приложения****

**Количество строк программного кода**:

Примерное количество строк кода приложения — 180 строк. Основные части кода:

* Логика игры (обработка ходов, проверка победителей).
* Графический интерфейс с использованием Tkinter.
* Реализация алгоритма Минимакс.

**Структуры данных**:

* **Двумерный список (матрица 3x3)** для представления игрового поля.
* **Кнопки** (объекты Tkinter) для отображения клеток на экране и получения ввода от пользователя.

**Алгоритмы**:

* **Минимакс** — для реализации игры с компьютером.
* **Проверка победителя** — алгоритм для проверки на наличие победной комбинации (горизонтальной, вертикальной или диагональной).
* **Проверка на ничью** — проверка, заполнилось ли всё поле, но победитель не найден.

**Используемые библиотеки**:

* **Tkinter** — для создания интерфейса, обработки событий и отображения игры.
* **math** — для выполнения математических операций (например, при реализации алгоритма Минимакс).

**Внешний вид приложения**:

* Игровое поле отображается в виде 3x3 сетки с кнопками, которые могут быть нажаты игроками.
* При завершении игры (победа или ничья) показывается сообщение с результатом, и игра предлагает начать заново.

**Средства контроля корректности ввода/вывода**:

* Ввод осуществляется через кнопки графического интерфейса, что исключает ошибки ввода с клавиатуры.
* Вывод информации об игре (победитель или ничья) осуществляется через окна сообщений Tkinter.

**2.2. Особенности реализации приложения**

**Структуры данных**:

**Матрица 3x3**: для представления игрового поля используется двумерный список:

field = [['' for \_ in range(3)] for \_ in range(3)]

Этот список обновляется каждый раз, когда игрок или компьютер делают ход. Каждый элемент списка может содержать значения "X", "O" или быть пустым (''), что позволяет легко отслеживать состояние игры.

**Альтернативные варианты**:

Можно было бы использовать одномерный список из 9 элементов вместо двумерного массива. Однако использование двумерного списка улучшает читаемость и упрощает реализацию проверок победителя.

**Алгоритм Минимакс**:

Алгоритм Минимакс реализован рекурсивно, где для каждого возможного хода вычисляется оценка состояния игры, и выбирается ход, который максимизирует шансы на победу для компьютера или минимизирует для игрока.

**Обоснование выбора**:

Выбор двумерного списка для игрового поля был сделан для удобства реализации и читаемости кода. Это также позволяет легко проверять выигрышные линии (горизонтальные, вертикальные и диагональные).

Алгоритм Минимакс был выбран для реализации ИИ, так как он подходит для игр с полным перебором возможных состояний и небольшим числом возможных ходов (игровое поле 3x3).

### **3. Обращение к программе**

#### ****Методы и алгоритмы****

**Основные методы приложения:**

**check\_winner()**: Проверка, есть ли победитель.

**check\_draw()**: Проверка, завершилась ли игра ничьей.

**player\_turn()**: Логика хода игрока.

**computer\_turn()**: Логика хода компьютера, использующего алгоритм Минимакс.

**minimax()**: Реализация алгоритма Минимакс для вычисления оптимального хода для компьютера.

**start\_new\_game()**: Функция для начала новой игры.

**game\_over()**: Функция для завершения игры и вывода результата.

**Используемые библиотеки**:

* **Tkinter**: для создания GUI и обработки пользовательских событий. Основные используемые компоненты: Tk(), Button(), Label().
* **Button**: используется для создания кнопок на игровом поле.
* **Label**: используется для отображения сообщений о победителе или ничьей.

### **4. Сообщения**

Перечень сообщений, которые могут быть выведены при завершении игры:

1. **Победа игрока**:

"Игрок (X) победил!" — если игрок, играющий за крестики, выигрывает.

"Игрок (O) победил!" — если игрок, играющий за нолики, выигрывает.

2. **Победа компьютера**:

"Компьютер (X) победил!" — если компьютер, играющий за крестики, выигрывает.

"Компьютер (O) победил!" — если компьютер, играющий за нолики, выигрывает.

3. **Ничья**:

"Ничья!" — если игра заканчивается ничьей.

4. **При запуске новой игры**:

"Новая игра!" — при старте новой игры после завершения предыдущей.