МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Измерительно-вычислительные комплексы»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | |  | |  |  |
| |  |  | | --- | --- | | ***Подп. и дата*** |  | | ***Инв. № дубл.*** |  | | ***Взам. инв. №*** |  | | ***Подп. и дата*** |  | | ***Инв. № подл*** |  | |  | | **Пояснительная записка**  на лабораторную работу №8  по дисциплине «Алгоритмы и структуры данных»  **Тема** «Разработка компьютерной игры Крестики Нолики» | | | | | | |
|  | | |  | |  | | | |
|  | | Исполнитель  студент гр. ИСТбд-21  Казаров Д.С.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | |
|  | | 2024 | | | | | | |

**Введение**

Разрабатываемое игровое приложение называется "Крестики-нолики" и представляет собой классическую настольную игру, в которой два игрока поочередно размещают крестики и нолики на поле размером 3x3. Цель игры — первым собрать три своих символа в ряд по горизонтали, вертикали или диагонали. Приложение разработано с использованием библиотеки Tkinter и языка программирования Python.

Выбор данной технологии обусловлен простотой реализации пользовательского интерфейса с помощью Tkinter, а также гибкостью Python, что позволяет создавать компактные, но эффективные программы. Приложение демонстрирует навыки программирования, разработки графического интерфейса и алгоритмов, а также взаимодействие с пользователем через событийное управление.

**1. Проектная часть**

**1.1. Постановка задачи на разработку приложения**

Определяется в рамках задания на курсовую работу. Игра предназначена для двух игроков и для игры с ИИ, которые поочередно ходят, вводя свои символы ("X" и "O") на игровое поле. В игре предусмотрено обнаружение победных условий и возможность перезапуска.

**1.2. Математические методы**

В программе используется простая логическая проверка на наличие трех одинаковых символов в ряд по горизонтали, вертикали или диагонали для определения победителя. Все возможные комбинации выигрышных ситуаций фиксируются и проверяются на каждом ходе игрока.

Основные математические операции:

* Проверка строк и столбцов на наличие одинаковых элементов.
* Проверка двух диагоналей игрового поля.

Этот подход минималистичен и не требует сложных вычислений, что делает его подходящим для настольных игр такого типа.

**1.3. Архитектура и алгоритмы**

**1.3.1. Архитектура**

Архитектура программы включает следующие компоненты:

* **Игровое поле** — матрица 3x3, которая хранит текущее состояние ячеек: пусто, "X" или "O".
* **Игровая логика** — набор функций, которые проверяют правильность хода, определяют победителя и отслеживают ничейные ситуации.
* **Графический интерфейс** — реализован с использованием библиотеки Tkinter. Представляет собой сетку из кнопок, которые обновляются в зависимости от ходов игроков.

**1.3.2. Алгоритм игры**

Алгоритм игры состоит из нескольких этапов:

1. **Инициализация**: создается пустое игровое поле.
2. **Ход игрока**: при нажатии на ячейку происходит проверка доступности клетки для хода. Если клетка пуста, она заполняется символом текущего игрока.
3. **Проверка победы**: после каждого хода проверяются все возможные выигрышные комбинации (строки, столбцы, диагонали).
4. **Конец игры**: если победитель найден или все ячейки заполнены (ничья), выводится сообщение о результате. Игрокам предлагается начать новую игру.

**1.3.3. Алгоритм проверки победы**

1. Программа проверяет три строки, три столбца и две диагонали на наличие одинаковых символов.
2. Если найдена последовательность из трех одинаковых символов, игра завершена с победой соответствующего игрока.
3. Если все клетки заполнены, а победитель не определен, игра завершается ничьей.

**1.4. Тестирование**

**1.4.1. Описание отчета о тестировании**

Отчет о тестировании включает проверку функциональности каждого компонента игры. Особое внимание уделяется корректности определения победителя и реализации логики ничьей.

**1.4.2. Цель тестирования**

Цель тестирования — проверить, что игра корректно работает при различных сценариях (победа одного из игроков, ничья, перезапуск игры). Также проверяется устойчивость программы к неправильным действиям со стороны пользователя.

**1.4.3. Методика тестирования**

Используется метод черного ящика. Проверялись различные игровые сценарии: победы, ничьи, ввод некорректных данных, перезапуск игры.

**1.4.4. Проведенные тесты**

1. Тестирование правильности отображения ходов на игровом поле.
2. Тестирование на наличие победных комбинаций после каждого хода.
3. Тестирование ситуации ничьей (заполнение всех ячеек без победителя).
4. Тестирование корректной работы бота.

**1.4.5. Выводы**

По результатам тестирования выявлены и устранены ошибки, связанные с некорректным перезапуском игры и отображением информации о результате. В текущей версии все функции работают стабильно.

**2. Источники, использованные при разработке**

1. В.В. Шишкин, Д.С. Афонин. Разработка логических компьютерных игр с графическим интерфейсом в среде Python.
2. "Документация Tkinter" [Электронный ресурс] // Официальный сайт Python: <https://docs.python.org/3/library/tkinter.html>.
3. ГОСТ 19.701-90 «Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения». – М.: Издательство стандартов, 1990.
4. В.В. Шишкин, Д.С. Афонин РАЗРАБОТКА ЛОГИЧЕСКИХ КОМПЬЮТЕРНЫХ ИГР С ГРАФИЧЕСКИМ ИНТЕРФЕЙСОМ В СРЕДЕ ПИТОН