**Комплект программной документации**

**1. Техническое задание (ТЗ)**

**1.1 Наименование программы**

"Крестики-нолики с искусственным интеллектом (ИИ)"

**1.2 Назначение программы**

Программа предназначена для игры в "Крестики-нолики" на поле 3x3 с графическим интерфейсом. Пользователь может выбирать сторону (X или O) и играть против бота, который использует алгоритмы для принятия оптимальных решений.

**1.3 Основание для разработки**

Разработка ведется в рамках учебного задания для изучения программирования и алгоритмов на языке Python.

**2. Требования к программе**

**2.1 Требования к функционалу**

* Возможность выбора стороны (X или O).
* Графический интерфейс для управления игрой.
* Игра с ботом, который принимает оптимальные решения.
* Проверка правил игры: определение победителя, ничьей.
* Автоматический перезапуск после завершения партии.

**2.2 Требования к надежности**

Программа должна корректно обрабатывать некорректные действия пользователя (например, попытки ходить в занятые клетки).

**2.3 Интерфейс программы**

* Использование библиотеки Tkinter для создания графического интерфейса.
* Удобное управление с помощью мыши.

**2.4 Условия эксплуатации**

* Программа предназначена для запуска на Windows, macOS и Linux.
* Требуется установленный интерпретатор Python версии 3.7 и выше.

**3. Этапы разработки и тестирования**

1. Разработка алгоритма игры и интерфейса.
2. Реализация искусственного интеллекта для игры бота.
3. Тестирование всех сценариев игры, включая пограничные случаи.
4. Подготовка документации.

**2. Пояснительная записка**

**2.1 Описание программы**

Программа реализует игру "Крестики-нолики" с использованием языка Python и библиотеки Tkinter для графического интерфейса. Основные компоненты программы:

1. **Игровое поле**: Представлено массивом из 9 элементов (3x3). Каждый элемент может содержать X, O или быть пустым.
2. **Графический интерфейс**: Кнопки для каждой клетки поля и элементы для управления игрой.
3. **Алгоритм бота**: Использует стратегию на основе алгоритма минимакс для принятия оптимальных решений.

**2.2 Используемые технологии**

* **Язык программирования**: Python 3.7+
* **Графический интерфейс**: Tkinter
* **Алгоритмы**: Минимакс для искусственного интеллекта.

**2.3 Архитектура программы**

* Основной класс: TicTacToe, содержащий логику игры, пользовательский интерфейс и взаимодействие с ботом.
* Методы класса:
  + make\_move: Обработка хода игрока.
  + bot\_move: Генерация хода бота.
  + check\_winner: Проверка состояния игры (победа, ничья).
  + best\_move: Логика выбора лучшего хода для бота.

**3. Руководство программиста**

**3.1 Установка и запуск программы**

1. Убедитесь, что установлен Python версии 3.7 или выше.
2. Установите библиотеку Tkinter (обычно входит в стандартный пакет Python).
3. Скачайте исходный код программы.
4. Запустите файл с игрой командой:

python tic\_tac\_toe.py

**3.2 Структура кода**

Код организован в виде единого класса TicTacToe. Важные методы:

* \_\_init\_\_: Инициализация игрового интерфейса.
* make\_move: Логика обработки хода игрока.
* bot\_move: Логика хода бота.
* check\_winner: Проверка выигрыша или ничьей.

**3.3 Расширение функциональности**

Для расширения функционала можно:

* Изменить размер игрового поля (требуется изменение логики check\_winner и интерфейса).
* Добавить режим игры между двумя игроками.
* Внедрить более сложный алгоритм ИИ.

**4. Тестовая документация**

**4.1 Mind Map**

**Ключевые элементы для тестирования:**

* Проверка правильности работы интерфейса (нажатие кнопок, выбор стороны).
* Проверка логики определения победителя.
* Тестирование работы бота в различных сценариях.
* Обработка некорректных действий (например, повторный ход на занятую клетку).

**4.2 Чек-лист**

|  |  |  |
| --- | --- | --- |
| **ID** | **Проверка** | **Ожидаемый результат** |
| 1 | Запуск программы | Интерфейс загружается |
| 2 | Выбор стороны (X или O) | Сторона выбирается, игра начинается |
| 3 | Нажатие на клетку | Символ игрока отображается |
| 4 | Победная комбинация игрока | Выводится сообщение о победе игрока |
| 5 | Победная комбинация бота | Выводится сообщение о победе бота |
| 6 | Ничья | Выводится сообщение о ничьей |
| 7 | Ход на занятую клетку | Ничего не происходит |
| 8 | Бот блокирует возможную победу игрока | Ход бота предотвращает победу игрока |

**4.3 Набор тест-кейсов**

**Тест-кейс 1: Проверка выигрыша игрока**

* **Предусловие**: Поле почти заполнено, игрок может победить.
* **Шаги**:
  1. Выбрать сторону (например, X).
  2. Ходить так, чтобы создать победную комбинацию.
* **Ожидаемый результат**: Сообщение о победе игрока.

**Тест-кейс 2: Проверка блокировки игрока**

* **Предусловие**: У игрока есть возможность победить на следующем ходу.
* **Шаги**:
  1. Создать ситуацию, где игрок может выиграть.
  2. Проверить ход бота.
* **Ожидаемый результат**: Бот блокирует выигрыш игрока.

**Тест-кейс 3: Проверка ничьей**

* **Предусловие**: Поле заполнено полностью, победных комбинаций нет.
* **Шаги**:
  1. Последовательно заполнить поле.
* **Ожидаемый результат**: Сообщение о ничьей.