**1. Структуры данных**

В игре "Крестики-нолики" используются следующие основные структуры данных:

**1.1 Игровое поле (board):**

* Представлено в виде двумерного списка размером 3x3.
* Каждая ячейка может содержать одно из значений:
  + ' ' — пустая клетка.
  + 'X' — крестик, занятый игроком.
  + 'O' — нолик, занятый компьютером.

Пример игрового поля:

board = [

['X', ' ', 'O'],

[' ', 'X', ' '],

['O', ' ', ' ']

]

**1.2 Кнопки интерфейса (buttons):**

* Используется двумерный список buttons для хранения кнопок tkinter, которые отображают текущее состояние игрового поля и обрабатывают взаимодействия игрока.

**1.3 Игроки:**

* В коде игроки представлены символами:
  + Игрок: 'X'.
  + Компьютер: 'O'.

**1.4 Переменные состояния:**

* **board** — хранит текущее состояние игрового поля.
* **current\_player** — отвечает за текущего игрока. Смена игрока происходит в ходе игры.

**2. Алгоритмы**

**2.1 Алгоритм хода игрока (player\_move):**

* Отвечает за обработку хода игрока.
* Последовательность действий:
  1. Проверить, что выбранная клетка свободна.
  2. Установить символ текущего игрока ('X').
  3. Проверить, привёл ли ход к победе или ничьей:
     + Если победа — вывести сообщение о выигрыше и отключить кнопки.
     + Если ничья — вывести сообщение о ничьей и отключить кнопки.
  4. Передать ход компьютеру, если игра не завершена.

**2.2 Алгоритм ИИ (компьютерный ход, computer\_move):**

* Компьютер использует алгоритм **minimax** для определения оптимального хода.
* Логика:
  1. Перебрать все возможные ходы на пустых клетках.
  2. Для каждого хода вызвать функцию minimax, оценивающую результат.
  3. Выбрать ход с наивысшей оценкой.
  4. Сделать ход и проверить победу или ничью.

**2.3 Алгоритм оценки позиции (minimax):**

* Рекурсивный алгоритм для перебора всех возможных ходов и выбора наилучшего.
* Оценка:
  + Победа компьютера ('O') возвращает 1.
  + Победа игрока ('X') возвращает -1.
  + Ничья возвращает 0.

**2.4 Алгоритм проверки победы (check\_winner):**

* Проверяет, выиграл ли текущий игрок. Логика:
  1. Проверка всех строк.
  2. Проверка всех столбцов.
  3. Проверка двух диагоналей.

**2.5 Алгоритм проверки ничьей (is\_full):**

* Возвращает True, если все клетки заполнены (нет пустых клеток ' '), иначе False.

**2.6 Алгоритм сброса игры (reset\_game):**

* Полностью обнуляет игру:
  + Очищает массив board.
  + Возвращает кнопки интерфейса в начальное состояние.
  + Сбрасывает статус.

**3. Графический интерфейс**

**3.1 Отображение игрового поля:**

* Игровое поле представлено в виде сетки из кнопок tkinter.
* Каждая кнопка отображает содержимое соответствующей клетки board:
  + Пустая клетка — кнопка доступна для взаимодействия.
  + Занятая клетка ('X' или 'O') — кнопка отключается.

**3.2 Отображение статуса игры:**

* Результаты игры (победа, ничья) выводятся в текстовом элементе интерфейса (result\_label).

**3.3 Кнопка сброса:**

* Кнопка reset\_button позволяет начать игру заново, очищая игровое поле и статус.

**4. Итеративность проектирования**

Проект поддерживает добавление новых функций и улучшений:

* Возможность отмены хода может быть реализована через сохранение предыдущих состояний board.
* Добавление подсказок для игрока возможно путём вызова minimax для оценки оптимальных ходов.
* В дальнейшем интерфейс может быть расширен для учёта игры на большем поле или для игры двух человек.