**Федеральное государственное автономное образовательное учреждение**

**высшего образования**

**Национальный исследовательский университет**

**"Высшая школа экономики"**

**Московский институт электроники и математики им. А.Н. Тихонова**

**Департамент компьютерной инженерии**

**Руководство пользователя**

**“Игра в крестики-нолики с искусственным интеллектом на языке программирования C”**

**Выполнил студент:**

Кучерявенков Тимур Сергеевич, БИВ243

Москва 2025

**Структура программы.**

**Программа состоит из двух файлов:** gomoku\_fixed.c и difficulty\_gui.c.

**Файлы состоят из следующих функций:**

 **in\_field** – Проверяет, находятся ли координаты клетки (r, c) внутри границ доски N × N и возвращает true, если ход допустим. Применяется всеми вспомогательными процедурами для фильтрации выходов за пределы.

 **win** – Перебирает каждую занятую клетку и ищет непрерывную последовательность длиной W по горизонтали, вертикали или обеим диагоналям. Возвращает true, как только обнаруживает победную линию для заданного символа.

 **board\_full** – Линейно сканирует массив доски и определяет, остались ли свободные клетки. Если пустых клеток нет, сообщает о заполненном поле, что сигнализирует о ничьей.

 **score\_position** – Вызывает win для символов компьютера и игрока; возвращает +1000, если выигрывает компьютер, –1000, если выигрывает пользователь, или 0, если явного победителя ещё нет. Это простая эвристика для алгоритма Minimax.

 **gen\_moves** – Формирует список до 200 возможных ходов: собирает все пустые клетки, смежные хотя бы с одной занятой, либо первую попавшуюся пустую, если доска пуста. Сокращает ветвление дерева поиска до «разумных» вариантов.

 **minimax** – Рекурсивно анализирует позиции на заданную глубину (1 или 3), применяя альфа/бета-отсечения. Для каждой ветви возвращает оценку, выбирая максимум для компьютера и минимум для игрока.

 **compute\_best\_move** – Публичная точка входа: принимает текущую доску, параметры партии и уровень сложности, вызывает gen\_moves и minimax, выбирает ход с лучшей оценкой; на простом уровне (diff = 1) берёт случайную свободную клетку. Возвращает структуру Move с координатами найденного хода.

**Подробное описание функций:**

**compute\_best\_move -** получает текущую доску, параметры партии и уровень сложности, чтобы выбрать координаты следующего хода. Сначала она копирует данные в глобальные переменные и через gen\_moves формирует список пустых клеток рядом с занятыми. При сложности 1 ход выбирается случайно; при сложностях 2-3 для каждой клетки запускается minimax (глубина 1 или 3) с α/β-отсечениями, временно меняя роли, если ход ищется за игрока. Лучшая оценка сохраняет координаты оптимального хода. Итоговая структура Move с этими координатами возвращается вызывающей стороне.

**Пример использования**

Создайте массив char board[50][50], заполните его пробелами и расставьте уже сделанные ходы.  
Вызовите Move m = compute\_best\_move(15, 5, board, 'x', 'o', 3, 'o'); — функция вернёт оптимальные координаты для символа «o» на поле 15 × 15 при победной линии 5 и максимальной сложности.  
Далее выполните board[m.r][m.c] = 'o';, выведите доску и продолжайте игру.