Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

Специальность 09.02.07 Информационные системы и программирование

Квалификация: Программист

Практическая работа №1

По дисциплине «МДК 04.01 Внедрение и поддержка компьютерных систем»

Листов:

|  |  |
| --- | --- |
| Выполнила:  Петриашвили М.З.  Студент группы: П50-6-22 | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_\_А.А. Никонова  «\_\_\_\_» \_\_\_\_\_\_\_\_ 2025 года |

Москва 2025

Цель работы: сделать игру крестики-нолики в консоли с выбором игры на языке программирования dart в visual studio code.

Ход работы:

1. В данной практической работе были реализованы следующие методы, выполняющие свои определенные функции:
   1. Функция reset()  
      Сбрасывает игровое поле, создавая пустую матрицу размером *n × n*. Используется при начале новой партии.
   2. Функция printBoard()  
      Выводит текущее состояние игрового поля на экран. Для удобства отображаются номера строк и столбцов. Пустые клетки заменяются точкой, а занятые — символами игроков (X или O).
   3. Функция makeMove(r, c, sym)  
      Выполняет ход игрока в указанную клетку. Проверяет корректность координат и то, что клетка свободна. Если ход возможен, ставит символ игрока и возвращает подтверждение, иначе отклоняет ход.
   4. Функция isWin(sym)  
      Проверяет, выиграл ли игрок с указанным символом. Для этого анализируются все строки, столбцы и диагонали. Если вся линия заполнена одним символом, объявляется победа.
   5. Функция isDraw()  
      Определяет ситуацию ничьей. Если на поле нет пустых клеток и ни один из игроков не выиграл, игра завершается ничьей.
   6. Функция availableMoves()  
      Формирует список всех свободных клеток игрового поля, в которые можно выполнить ход. Используется искусственным интеллектом для анализа возможных действий.
   7. Функция evaluate(ai, human)  
      Оценивает текущее положение на поле. Если победил компьютер, возвращает положительное значение, если победил человек — отрицательное, иначе ноль.
   8. Функция heuristicScore(ai, human)  
      Вычисляет приблизительную силу позиции. Чем больше символов игрока в одной линии, тем выше её ценность. Если линия заблокирована соперником, она не учитывается. Эта функция используется для оценки позиций, когда игра ещё не завершена.
   9. Функция minimax(...)  
      Реализует алгоритм минимакс с отсечением по альфа-бета. Перебирает возможные ходы, имитируя игру за обоих игроков. Алгоритм максимизирует выигрыш компьютера и минимизирует выигрыш человека. Для оптимизации используется ограничение глубины поиска.
   10. . Функция getBestMove(ai, human)  
       Определяет лучший возможный ход для компьютера. Сначала проверяет возможность немедленной победы или необходимость блокировки соперника, затем выбирает оптимальный вариант с помощью алгоритма минимакс.
   11. Функция readInt(prompt, min, max)  
       Обеспечивает безопасный ввод числовых данных пользователем. Проверяет корректность формата и принадлежность числа указанному диапазону. При ошибках запрашивает повторный ввод.
   12. Функция main()  
       Главная функция программы. Организует процесс игры: приветствие, выбор размера поля и режима, случайное определение первого игрока. В цикле поочерёдно выполняются ходы игроков (или игрока и компьютера), проверяется окончание игры и выводится результат. После завершения предлагается сыграть новую партию.
2. Код работы:

import 'dart:io';

import 'dart:math';

class TicTacToe {

  int n;

  List<List<String>> board;

  String playerSymbol = 'X';

  String aiSymbol = 'O';

  Random rand = Random();

  TicTacToe(this.n) : board = List.generate(n, (\_) => List.filled(n, ' '));

  void reset() {

    board = List.generate(n, (\_) => List.filled(n, ' '));

  }

  void printBoard() {

    print('\n   ' + List.generate(n, (i) => (i + 1).toString().padLeft(2)).join(' '));

    for (int i = 0; i < n; i++) {

      String row = board[i].map((c) => c == ' ' ? '.' : c).join('  ');

      print((i + 1).toString().padLeft(2) + ' ' + row);

    }

    print('');

  }

  bool makeMove(int r, int c, String sym) {

    if (r < 0 || r >= n || c < 0 || c >= n) return false;

    if (board[r][c] != ' ') return false;

    board[r][c] = sym;

    return true;

  }

  bool isWin(String sym) {

    for (int i = 0; i < n; i++) {

      bool all = true;

      for (int j = 0; j < n; j++) {

        if (board[i][j] != sym) { all = false; break; }

      }

      if (all) return true;

    }

    for (int j = 0; j < n; j++) {

      bool all = true;

      for (int i = 0; i < n; i++) {

        if (board[i][j] != sym) { all = false; break; }

      }

      if (all) return true;

    }

    bool all = true;

    for (int i = 0; i < n; i++) {

      if (board[i][i] != sym) { all = false; break; }

    }

    if (all) return true;

    all = true;

    for (int i = 0; i < n; i++) {

      if (board[i][n - 1 - i] != sym) { all = false; break; }

    }

    if (all) return true;

    return false;

  }

  bool isDraw() {

    for (var row in board) {

      if (row.contains(' ')) return false;

    }

    return true;

  }

  List<List<int>> availableMoves() {

    List<List<int>> mv = [];

    for (int i = 0; i < n; i++) {

      for (int j = 0; j < n; j++) {

        if (board[i][j] == ' ') mv.add([i, j]);

      }

    }

    return mv;

  }

  int evaluate(String ai, String human) {

    if (isWin(ai)) return 1000;

    if (isWin(human)) return -1000;

    return 0;

  }

  int heuristicScore(String ai, String human) {

    int score = 0;

    for (int i = 0; i < n; i++) {

      bool aiBlock = false, humanBlock = false;

      int aiCount = 0, humanCount = 0;

      for (int j = 0; j < n; j++) {

        if (board[i][j] == ai) aiCount++;

        if (board[i][j] == human) humanCount++;

      }

      if (humanCount == 0 && aiCount > 0) score += pow(10, aiCount).toInt();

      if (aiCount == 0 && humanCount > 0) score -= pow(10, humanCount).toInt();

    }

    for (int j = 0; j < n; j++) {

      int aiCount = 0, humanCount = 0;

      for (int i = 0; i < n; i++) {

        if (board[i][j] == ai) aiCount++;

        if (board[i][j] == human) humanCount++;

      }

      if (humanCount == 0 && aiCount > 0) score += pow(10, aiCount).toInt();

      if (aiCount == 0 && humanCount > 0) score -= pow(10, humanCount).toInt();

    }

    int aiCount = 0, humanCount = 0;

    for (int i = 0; i < n; i++) {

      if (board[i][i] == ai) aiCount++;

      if (board[i][i] == human) humanCount++;

    }

    if (humanCount == 0 && aiCount > 0) score += pow(10, aiCount).toInt();

    if (aiCount == 0 && humanCount > 0) score -= pow(10, humanCount).toInt();

    aiCount = 0; humanCount = 0;

    for (int i = 0; i < n; i++) {

      if (board[i][n - 1 - i] == ai) aiCount++;

      if (board[i][n - 1 - i] == human) humanCount++;

    }

    if (humanCount == 0 && aiCount > 0) score += pow(10, aiCount).toInt();

    if (aiCount == 0 && humanCount > 0) score -= pow(10, humanCount).toInt();

    return score;

  }

  int minimax(int depth, bool isMax, int alpha, int beta, String ai, String human, int depthLimit) {

    int eval = evaluate(ai, human);

    if (eval.abs() >= 1000 || depth == depthLimit || isDraw()) {

      if (eval.abs() >= 1000) {

        return eval + (isMax ? -depth : depth);

      } else {

        return heuristicScore(ai, human);

      }

    }

    if (isMax) {

      int maxEval = -1000000000;

      for (var mv in availableMoves()) {

        board[mv[0]][mv[1]] = ai;

        int res = minimax(depth + 1, false, alpha, beta, ai, human, depthLimit);

        board[mv[0]][mv[1]] = ' ';

        if (res > maxEval) maxEval = res;

        if (res > alpha) alpha = res;

        if (beta <= alpha) break;

      }

      return maxEval;

    } else {

      int minEval = 1000000000;

      for (var mv in availableMoves()) {

        board[mv[0]][mv[1]] = human;

        int res = minimax(depth + 1, true, alpha, beta, ai, human, depthLimit);

        board[mv[0]][mv[1]] = ' ';

        if (res < minEval) minEval = res;

        if (res < beta) beta = res;

        if (beta <= alpha) break;

      }

      return minEval;

    }

  }

  List<int> getBestMove(String ai, String human) {

    var moves = availableMoves();

    if (moves.isEmpty) return [-1, -1];

    int depthLimit = (n <= 3) ? 9 : (n == 4 ? 6 : 4);

    for (var mv in moves) {

      board[mv[0]][mv[1]] = ai;

      if (isWin(ai)) {

        board[mv[0]][mv[1]] = ' ';

        return mv;

      }

      board[mv[0]][mv[1]] = ' ';

    }

    for (var mv in moves) {

      board[mv[0]][mv[1]] = human;

      if (isWin(human)) {

        board[mv[0]][mv[1]] = ' ';

        return mv;

      }

      board[mv[0]][mv[1]] = ' ';

    }

    int bestVal = -1000000000;

    List<int> bestMove = moves[rand.nextInt(moves.length)];

    for (var mv in moves) {

      board[mv[0]][mv[1]] = ai;

      int moveVal = minimax(0, false, -1000000000, 1000000000, ai, human, depthLimit);

      board[mv[0]][mv[1]] = ' ';

      if (moveVal > bestVal) {

        bestVal = moveVal;

        bestMove = mv;

      }

    }

    return bestMove;

  }

}

int readInt(String prompt, {int min = -999999, int max = 999999}) {

  while (true) {

    stdout.write(prompt);

    String? line = stdin.readLineSync();

    if (line == null) return min;

    int? val = int.tryParse(line.trim());

    if (val == null) {

      print('Ошибка: введите целое число.');

      continue;

    }

    if (val < min || val > max) {

      print('Ошибка: число должно быть в диапазоне $min..$max.');

      continue;

    }

    return val;

  }

}

void main() {

  print('=== Крестики-нолики ===');

  while (true) {

    int n = readInt('Введите размер игрового поля N (например 3): ', min: 3, max: 10);

    TicTacToe game = TicTacToe(n);

    print('Выберите режим:');

    print('1) Человек vs Человек');

    print('2) Человек vs Робот');

    int mode = readInt('Ваш выбор (1 или 2): ', min: 1, max: 2);

    Random rand = Random();

    bool humanFirst = rand.nextBool();

    print('\nСлучайный выбор: первым ходит ${humanFirst ? 'X (человек)' : (mode==2 ? 'Робот (X)' : 'Игрок 2 (X)')}.');

    String current = 'X';

    bool vsRobot = (mode == 2);

    String humanSymbol = 'X';

    String aiSymbol = 'O';

    if (vsRobot) {

      if (humanFirst) {

        humanSymbol = 'X';

        aiSymbol = 'O';

      } else {

        humanSymbol = 'O';

        aiSymbol = 'X';

      }

      game.playerSymbol = 'X';

      game.aiSymbol = 'O';

    }

    bool gameOver = false;

    game.printBoard();

    while (!gameOver) {

      if (vsRobot) {

        bool isHumanTurn;

        if (humanFirst) {

          isHumanTurn = (current == humanSymbol);

        } else {

          isHumanTurn = (current == humanSymbol);

        }

        if (isHumanTurn) {

          print('Ход человека ($current). Введите: строка столбец (через пробел).');

          bool moved = false;

          while (!moved) {

            stdout.write('> ');

            String? line = stdin.readLineSync();

            if (line == null) continue;

            var parts = line.trim().split(RegExp(r'\s+'));

            if (parts.length != 2) {

              print('Ошибка: введите два числа, например "1 1".');

              continue;

            }

            int? r = int.tryParse(parts[0]);

            int? c = int.tryParse(parts[1]);

            if (r == null || c == null) {

              print('Ошибка: неверный формат чисел.');

              continue;

            }

            r -= 1; c -= 1;

            if (!game.makeMove(r, c, current)) {

              print('Невозможный ход — клетка занята или вне поля.');

              continue;

            }

            moved = true;

          }

          game.printBoard();

          if (game.isWin(current)) {

            print('Победил игрок ($current)!');

            gameOver = true;

            break;

          } else if (game.isDraw()) {

            print('Ничья!');

            gameOver = true;

            break;

          } else {

            current = (current == 'X') ? 'O' : 'X';

            continue;

          }

        } else {

          print('Ход робота ($current)...');

          String aiSym = current;

          String humanSym = (current == 'X') ? 'O' : 'X';

          List<int> mv = game.getBestMove(aiSym, humanSym);

          if (mv[0] == -1) {

            var avail = game.availableMoves();

            mv = (avail.isNotEmpty) ? avail[rand.nextInt(avail.length)] : [-1, -1];

          }

          if (mv[0] != -1) {

            game.makeMove(mv[0], mv[1], current);

            print('Робот делает ход: ${mv[0] + 1} ${mv[1] + 1}');

          } else {

            print('Робот не нашёл ход — ничья?');

          }

          game.printBoard();

          if (game.isWin(current)) {

            print('Победил робот ($current)!');

            gameOver = true;

            break;

          } else if (game.isDraw()) {

            print('Ничья!');

            gameOver = true;

            break;

          } else {

            current = (current == 'X') ? 'O' : 'X';

            continue;

          }

        }

      } else {

        print('Ход игрока ($current). Введите: строка столбец (через пробел).');

        bool moved = false;

        while (!moved) {

          stdout.write('> ');

          String? line = stdin.readLineSync();

          if (line == null) continue;

          var parts = line.trim().split(RegExp(r'\s+'));

          if (parts.length != 2) {

            print('Ошибка: введите два числа, например "1 1".');

            continue;

          }

          int? r = int.tryParse(parts[0]);

          int? c = int.tryParse(parts[1]);

          if (r == null || c == null) {

            print('Ошибка: неверный формат чисел.');

            continue;

          }

          r -= 1; c -= 1;

          if (!game.makeMove(r, c, current)) {

            print('Невозможный ход — клетка занята или вне поля.');

            continue;

          }

          moved = true;

        }

        game.printBoard();

        if (game.isWin(current)) {

          print('Победил игрок ($current)!');

          gameOver = true;

          break;

        } else if (game.isDraw()) {

          print('Ничья!');

          gameOver = true;

          break;

        } else {

          current = (current == 'X') ? 'O' : 'X';

          continue;

        }

      }

    }

    stdout.write('\nЗапустить новую игру? (y/n): ');

    String? ans = stdin.readLineSync();

    if (ans == null || ans.toLowerCase() != 'y') {

      print('Выход. Спасибо за игру!');

      break;

    }

  }

}