Описание:

Для реализации игры в крестики-нолики с компьютером использовался алгоритм минимакс. Минимакс — это рекурсивный алгоритм, который используется для выбора оптимального хода для игрока, предполагая, что противник также играет оптимально. Как следует из названия, его цель - минимизировать максимальные потери (минимизировать сценарий наихудшего случая).

Код программы C#:

```
namespace TiTaTo
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            buttonsArr = new List<Button> { button1, button2, button3,
             button4, button5, button6, button7, button8, button9 };
            radioButton1.Checked = true;
        }
        List<Button> buttonsArr; //массив ячеек поля
        //обработчик нажатия на клетку поля
        private void button_click(object sender, EventArgs e)
        {
            Button button = (Button)sender;
            move(Convert.ToInt32(button.Tag), human);
        }
        //начальное состояние поля
        string[] board = { "0", "1", "2", "3", "4", "5", "6", "7", "8" };
        string human = "X"; //пользователь по умолчанию
        string comp = "O"; //компьютер по умолчанию
        int iter = 0;
        int roundNum = 0; //номер раунда
        class steps //объект хода
        {
            public string index;
            public int points;
        }
        //обработка хода
        public void move(int id, string player)
        {
            if (board[id] != "X" && board[id] != "O")
            {
                roundNum++;
                board[id] = player;
                buttonsArr[id].Enabled = false;
                buttonsArr[id].Text = human;

                if (winning(board, player))
                {
                    DialogResult result = MessageBox.Show("Победили ВЫ!",
                    "Вот так конец!");
                    if (result == DialogResult.OK) newGame();
                    return;
                }
                else if (roundNum > 8)
                {
```

```
                DialogResult result = MessageBox.Show("Да это же НИЧЬЯ!",
                 "Вот так конец!");
                if (result == DialogResult.OK) newGame();
                return;
            }
            else
            {
                roundNum++;
                var index = Convert.ToInt32(minimax(board, comp).index);
                buttonsArr[index].Enabled = false;
                buttonsArr[index].Text = comp;
                board[index] = comp;

                if (winning(board, comp))
                {
                    DialogResult result = MessageBox.Show("Компьютер
                     ПОБЕДИЛ!", "Вот так конец!");
                    if (result == DialogResult.OK) newGame();
                    return;
                }
                else if (roundNum == 0)
                {
                    DialogResult result = MessageBox.Show("Да это же
                     НИЧЬЯ!", "Вот так конец!");
                    if (result == DialogResult.OK) newGame();
                    return;
                }
            }
        }
    }
    void newGame() //очистка поля для новой игры
    {
        roundNum = 0; int i = 0;
        string[] clearBoard = { "0","1","2","3","4","5","6","7","8"};
        foreach (string r in clearBoard)
        {
            board[i] = r;
            i++;
        }
        foreach (var bb in buttonsArr)
        {
            bb.Enabled = true;
            bb.Text = "";
        }
    }
    //алгоритм минимакса
    steps minimax(string[] reboard, string player)
    {
        iter++;
        steps newStep = new steps();
        List<int> emptyCellsArr = (emptyCells(reboard)).ToList();
        if (winning(reboard, human))
        {
            newStep.points = -1;
            return newStep;
        }
        else if (winning(reboard, comp))
        {
            newStep.points = 1;
            return newStep;
        }
        else if (emptyCellsArr.Count == 0)
        {
```

```
            newStep.points = 0;
            return newStep;
        }
        List<steps> stepsArr = new List<steps>();
        for (var i = 0; i < emptyCellsArr.Count; i++)
        {
            steps step = new steps();
            step.index = reboard[emptyCellsArr[i]];
            reboard[emptyCellsArr[i]] = player;

            if (player == comp)
            {
                var minimaxValue = minimax(reboard, human);
                step.points = minimaxValue.points;
            }
            else
            {
                var minimaxValue = minimax(reboard, comp);
                step.points = minimaxValue.points;
            }
            reboard[emptyCellsArr[i]] = step.index;
            stepsArr.Add(step);
        }
        int bestStep = 0;
        if (player == comp)
        {
            var bestPoint = -1000;
            for (var i = 0; i < stepsArr.Count; i++)
            {
                if (stepsArr[i].points > bestPoint)
                {
                    bestPoint = stepsArr[i].points;
                    bestStep = i;
                }
            }
        }
        else
        {
            var bestPoint = 1000;
            for (var i = 0; i < stepsArr.Count; i++)
            {
                if (stepsArr[i].points < bestPoint)
                {
                    bestPoint = stepsArr[i].points;
                    bestStep = i;
                }
            }
        }
        return stepsArr[bestStep];
    }
    //метод нахождения пустых клеток
    List<int> emptyCells(string[] reboard)
    {
        List<int> emptyCellsArr = new List<int>();
        for (int i = 0; i < reboard.Length; i++)
        {
            if (reboard[i] != "X" && reboard[i] != "O")
             emptyCellsArr.Add(i);
        }
        return emptyCellsArr;
    }
    //проверка на соответствие выигрышным комбинациям
    bool winning(string[] board, string player)
```

```
        {
            if (
              (board[0] == player && board[1] == player && board[2] ==
               player) ||
              (board[3] == player && board[4] == player && board[5] ==
               player) ||
              (board[6] == player && board[7] == player && board[8] ==
               player) ||
              (board[0] == player && board[3] == player && board[6] ==
               player) ||
              (board[1] == player && board[4] == player && board[7] ==
               player) ||
              (board[2] == player && board[5] == player && board[8] ==
               player) ||
              (board[0] == player && board[4] == player && board[8] ==
               player) ||
              (board[2] == player && board[4] == player && board[6] ==
               player)
            ) return true;
            else return false;
        }
        //обработчик выбора X или O пользователем
        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {
            RadioButton radio = (RadioButton)sender;
            human = radio.Text;
            if (radio.Text.Contains("X")) comp = "O";
            else comp = "X";
        }

    }
}
```

Каждый ход обрабатывается методом move, в котором происходит вызов метода алгоритма минимакса, а также определяется победитель. Метод minimax содержит в себе алгоритм минимакса

Результатом работы алгоритма является выбор хода, обеспечивающего оптимальный результат. Алгоритм рассчитывает ходы до тех пор, пока не достигнет конца партии, будь то победа, поражение или ничья. В конечном состоянии, ИИ начислит себе положительное количество очков (+1) за победу, отрицательное (-1) — за поражение, и нейтральное (0) — за ничью.

В то же время алгоритм проводит аналогичные расчёты для ходов игрока. Он будет выбирать ход с наиболее высоким баллом, если ходит компьютер, и ход с наименьшим, если ходит игрок.

После окончания партии на экран выводится сообщением, содержащее информацию о победителе, проигравшем, либо ничье. Происходит сброс поля и ячеек в методе newGame.

На рис. 1 представлено окно программы при запуске. Поля пустые. Программа ожидает, когда пользователь сделает ход.

Чтобы запустить программу можно нажать на ярлык "Крестики-нолики", либо в папке "TiTaTo\bin\Debug\netcoreapp3.1\" запустить "TiTaTo.exe", либо открыть файл проекта в VS.

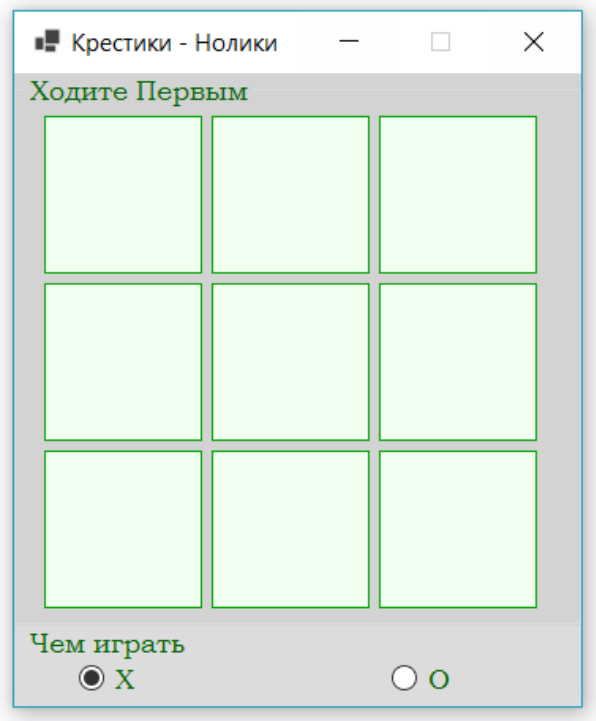

Рис. 1 Окно при запуске

На рис.2 пользователь сделал ход, поставив «X» в первую ячейку. Компьютер сделал ход «O» в пятую ячейку.

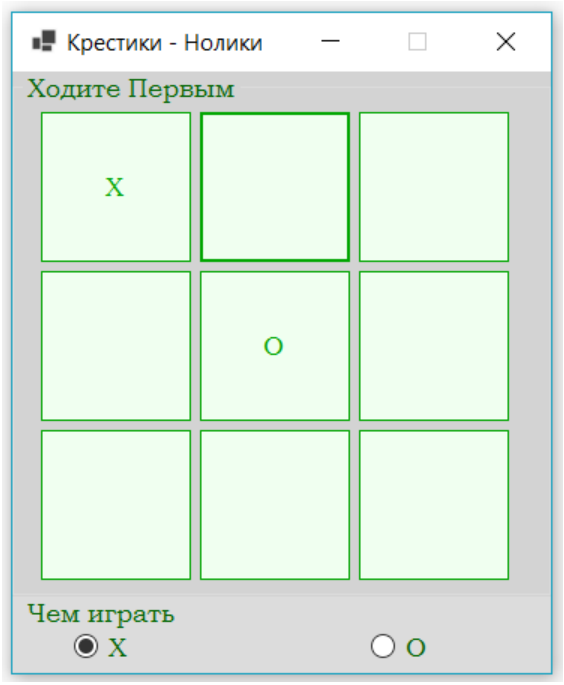

Рис. 2 Пользователь и компьютер сделали по одному ходу

На рис.3 представлена победа компьютера. Здесь пользователь ходил «X».

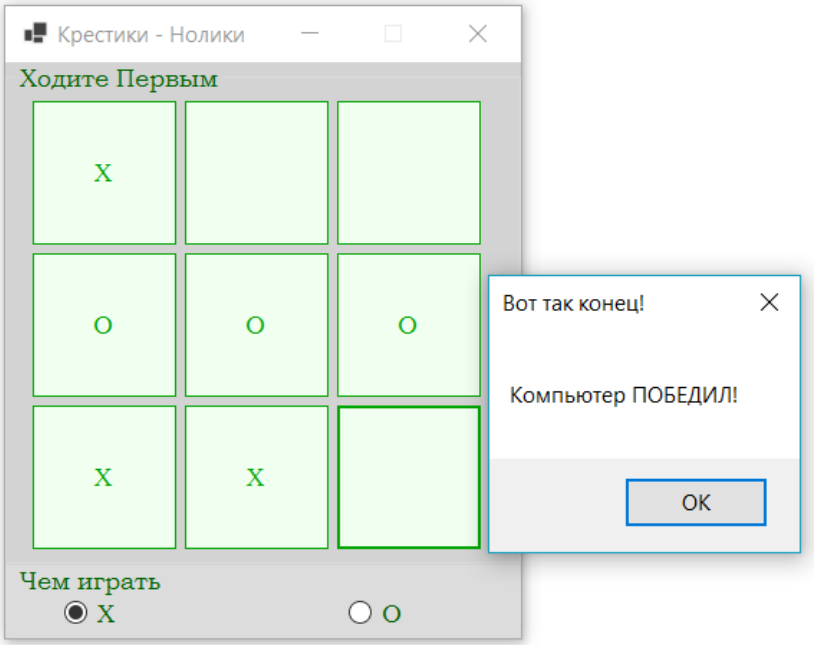

Рис. 3 Конец партии, где компьютер победил

На рис.4 представлена ничья. Здесь пользователь ходил за «O».

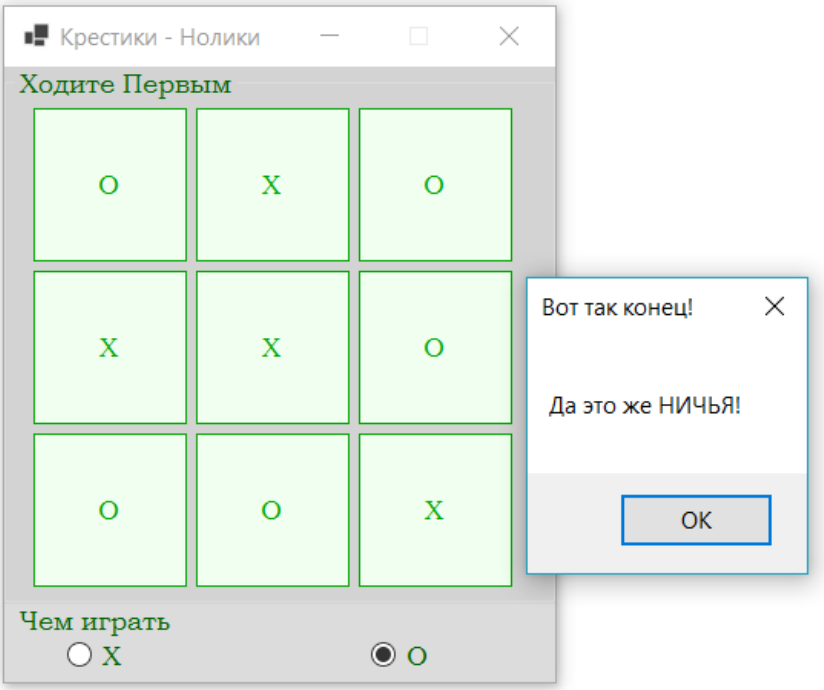

Рис. 4 Конец партии и ничья

Вывод: таким образом, использование алгоритма минимакс позволяет создать программу для игры Крестики-нолики с компьютером, который исключает возможность собственного проигрыша.