Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Магнитогорский государственный технический университет им. Г.И. Носова»

Многопрофильный колледж

Отделение №2 Информационные технологии и транспорт

**Отчёт**

по МДК.02.01. Технология разработки программного обеспечения

на тему: создание игры «Крестики нолики»

Исполнители: Пантелеев Александр, Рыбалка Евгений студенты группы ИСпПК-22-1

Преподаватель: Коржина В. С.

**Техническое задание**

Модули  
Вариант 14  
Тема «Крестики- нолики» -  
Составить программу, позволяющую играть на бесконечном поле в «крестики-нолики»:  
﻿﻿﻿Игроку с компьютером;  
﻿﻿﻿Двум игрокам.  
Если в качестве игрока выступает компьютер, программа делает первый ход. Делая очередной ход, программа анализирует ситуацию, рассчитывая возможные ходы противника на 1-2 хода, и в результате проведенного анализа поступает оптимальным образом.  
При решении задачи использовать модули, файлы, процедуры и Функции.

**Описание работы программы**

**1. Запуск программы**

* Инициализируется игровое поле (InitializeGame()).
* Создается сетка кнопок (InitializeGrid()).

**2. Начало игры**

* **Против компьютера** (PlayVsComputer\_Click):
  + Компьютер (O) ходит первым (MakeComputerMove).
* **Против игрока** (PlayVsPlayer\_Click):
  + Первый ход за X.

**3. Ход игрока**

1. Клик по кнопке → Button\_Click.
2. Если клетка свободна → MakeMove(row, col):
   * Запись хода (board + обновление кнопки).
   * Проверка победы (CheckWinner).
   * Если победы нет → переход хода (X ↔ O).
3. В режиме с ИИ → компьютер делает ход (MakeComputerMove).

**4. Логика компьютера (MakeComputerMove)**

1. **Пытается выиграть** (есть ли 2 O в ряд?).
2. **Блокирует игрока** (есть ли 2 X в ряд?).
3. **Ходит рядом со своими O** (стратегия).
4. **Случайный ход** (если нет вариантов).

**5. Проверка победы (CheckWinner)**

* Проверяет 4 направления (горизонталь, вертикаль, 2 диагонали).
* Если 3 в ряд → победа.

**6. Расширение поля**

* Если ход на границе → поле увеличивается (ExpandGrid).

**7. Дополнительно**

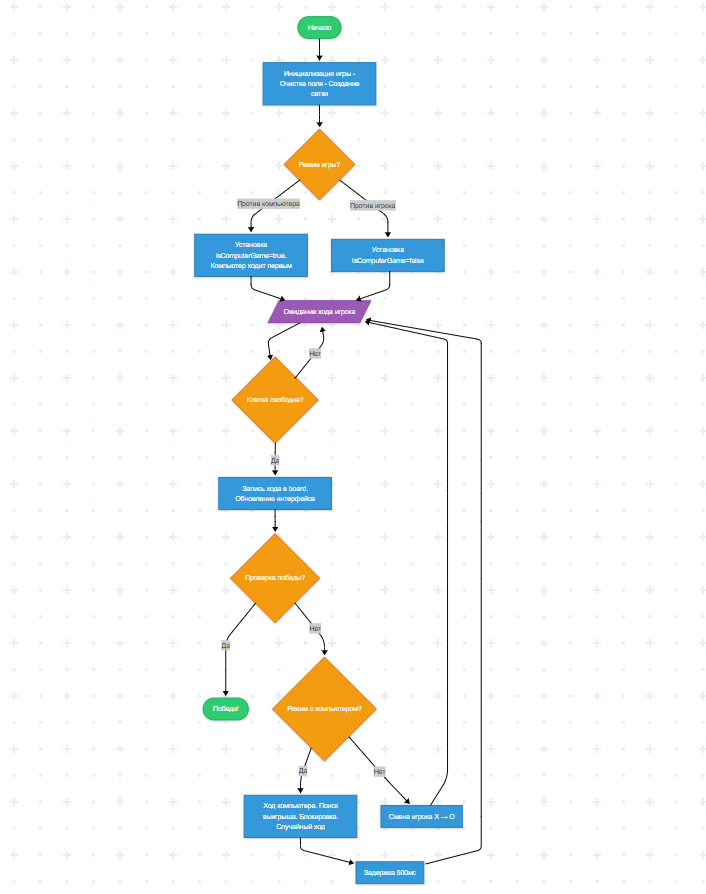
* **Меню**: Новая игра, правила, выход.
* **Статус**: Отображает текущего игрока.

**Ключевые моменты**

* **Поле бесконечно** (расширяется автоматически).
* **ИИ простой, но стратегический** (не случайный).
* **Победа проверяется динамически** (без жестких условий)

<https://github.com/Maninderi/workaem>

**Блок-схема**



**Приложение А**

**Листинг 1 (XAML-разметка главного окна)**

<Window x:Class="workaem.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:workaem"

mc:Ignorable="d"

Title="Крестики-нолики" Height="600" Width="800">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Menu Grid.Row="0">

<MenuItem Header="Игра">

<MenuItem Header="Новая игра" Click="NewGame\_Click"/>

<MenuItem Header="Против компьютера" Click="PlayVsComputer\_Click"/>

<MenuItem Header="Против игрока" Click="PlayVsPlayer\_Click"/>

<Separator/>

<MenuItem Header="Выход" Click="Exit\_Click"/>

</MenuItem>

<MenuItem Header="Справка">

<MenuItem Header="Правила игры" Click="Rules\_Click"/>

<MenuItem Header="О программе" Click="About\_Click"/>

</MenuItem>

</Menu>

<Grid Grid.Row="1" Margin="20">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<TextBlock x:Name="StatusText"

Text="Выберите режим игры в меню 'Игра'"

FontSize="20"

HorizontalAlignment="Center"

Margin="0,0,0,20"/>

<ScrollViewer Grid.Row="1"

HorizontalScrollBarVisibility="Auto"

VerticalScrollBarVisibility="Auto">

<Grid x:Name="GameGrid">

*<!-- Динамическое добавление кнопок -->*

</Grid>

</ScrollViewer>

</Grid>

</Grid>

<Window.Resources>

<Style TargetType="Button">

<Setter Property="FontSize" Value="48"/>

<Setter Property="FontWeight" Value="Bold"/>

<Setter Property="Margin" Value="2"/>

<Setter Property="Width" Value="60"/>

<Setter Property="Height" Value="60"/>

<Setter Property="Background" Value="White"/>

<Setter Property="BorderBrush" Value="#FF888888"/>

<Setter Property="BorderThickness" Value="1"/>

<Style.Triggers>

<Trigger Property="IsMouseOver" Value="True">

<Setter Property="Background" Value="#FFCCCCCC"/>

</Trigger>

<Trigger Property="IsPressed" Value="True">

<Setter Property="Background" Value="#FFAAAAAA"/>

</Trigger>

</Style.Triggers>

</Style>

</Window.Resources>

</Window>

**Листинг 2 (Инициализация игры и обработка меню)**

public partial class MainWindow : Window

{

private Dictionary<(int, int), string> board;

private bool isComputerGame;

private bool isGameActive;

private string currentPlayer;

private Random random;

private const int WIN\_LENGTH = 3;

private int minRow = -5, maxRow = 5;

private int minCol = -5, maxCol = 5;

public MainWindow()

{

InitializeComponent();

random = new Random();

InitializeGame();

}

private void InitializeGame()

{

board = new Dictionary<(int, int), string>();

isGameActive = false;

currentPlayer = "X";

InitializeGrid();

}

private void PlayVsComputer\_Click(object sender, RoutedEventArgs e)

{

InitializeGame();

isComputerGame = true;

isGameActive = true;

currentPlayer = "O"; *// Компьютер ходит первым*

UpdateStatus();

MakeComputerMove();

}

private void PlayVsPlayer\_Click(object sender, RoutedEventArgs e)

{

InitializeGame();

isComputerGame = false;

isGameActive = true;

UpdateStatus();

}

private void NewGame\_Click(object sender, RoutedEventArgs e)

{

if (isComputerGame)

PlayVsComputer\_Click(sender, e);

else

PlayVsPlayer\_Click(sender, e);

}

private void Exit\_Click(object sender, RoutedEventArgs e)

{

Close();

}

}

**Листинг 3 (Создание игрового поля и обработка ходов)**

private void InitializeGrid()

{

GameGrid.Children.Clear();

GameGrid.RowDefinitions.Clear();

GameGrid.ColumnDefinitions.Clear();

for (int i = minRow; i <= maxRow; i++)

{

GameGrid.RowDefinitions.Add(new RowDefinition());

GameGrid.ColumnDefinitions.Add(new ColumnDefinition());

}

for (int i = minRow; i <= maxRow; i++)

{

for (int j = minCol; j <= maxCol; j++)

{

Button button = new Button();

string rowName = i < 0 ? $"m{Math.Abs(i)}" : i.ToString();

string colName = j < 0 ? $"m{Math.Abs(j)}" : j.ToString();

button.Name = $"Button{rowName}\_{colName}";

button.Tag = new Tuple<int, int>(i, j);

button.Click += Button\_Click;

Grid.SetRow(button, i - minRow);

Grid.SetColumn(button, j - minCol);

GameGrid.Children.Add(button);

}

}

}

private async void Button\_Click(object sender, RoutedEventArgs e)

{

if (!isGameActive) return;

Button button = (Button)sender;

var coordinates = (Tuple<int, int>)button.Tag;

int row = coordinates.Item1;

int col = coordinates.Item2;

if (!board.ContainsKey((row, col)))

{

MakeMove(row, col);

if (isComputerGame && isGameActive)

{

await Task.Delay(500);

MakeComputerMove();

}

}

}

private void MakeMove(int row, int col)

{

board[(row, col)] = currentPlayer;

Button button = FindButton(row, col);

button.Content = currentPlayer;

if (CheckWinner(row, col))

{

MessageBox.Show($"Победил игрок {currentPlayer}!");

isGameActive = false;

return;

}

currentPlayer = currentPlayer == "X" ? "O" : "X";

UpdateStatus();

if (ShouldExpandGrid(row, col))

{

ExpandGrid();

}

}

**Листинг 4 (Логика ИИ компьютера)**

private void MakeComputerMove()

{

*// Поиск выигрышного хода*

foreach (var cell in GetEmptyCells())

{

board[cell] = "O";

if (CheckWinner(cell.Item1, cell.Item2))

{

MakeMove(cell.Item1, cell.Item2);

return;

}

board.Remove(cell);

}

*// Блокировка выигрыша игрока*

foreach (var cell in GetEmptyCells())

{

board[cell] = "X";

if (CheckWinner(cell.Item1, cell.Item2))

{

board.Remove(cell);

MakeMove(cell.Item1, cell.Item2);

return;

}

board.Remove(cell);

}

*// Ход рядом с существующими символами*

var nearMoves = GetNearMoves();

if (nearMoves.Count > 0)

{

var move = nearMoves[random.Next(nearMoves.Count)];

MakeMove(move.Item1, move.Item2);

return;

}

*// Случайный ход в центре*

int row = random.Next(-2, 3);

int col = random.Next(-2, 3);

while (board.ContainsKey((row, col)))

{

row = random.Next(-2, 3);

col = random.Next(-2, 3);

}

MakeMove(row, col);

}

**Листинг 5 (Проверка победы и вспомогательные методы)**

private bool CheckWinner(int row, int col)

{

var directions = new[] { (1, 0), (0, 1), (1, 1), (1, -1) };

foreach (var dir in directions)

{

int count = 1;

count += CountInDirection(row, col, dir.Item1, dir.Item2);

count += CountInDirection(row, col, -dir.Item1, -dir.Item2);

if (count >= WIN\_LENGTH) return true;

}

return false;

}

private int CountInDirection(int row, int col, int dRow, int dCol)

{

int count = 0;

int r = row + dRow;

int c = col + dCol;

string player = board[(row, col)];

while (board.TryGetValue((r, c), out string value) && value == player)

{

count++;

r += dRow;

c += dCol;

}

return count;

}

private Button FindButton(int row, int col)

{

return (Button)GameGrid.Children.Cast<UIElement>()

.FirstOrDefault(e => e is Button button &&

((Tuple<int, int>)button.Tag).Item1 == row &&

((Tuple<int, int>)button.Tag).Item2 == col);

}

private void UpdateStatus()

{

StatusText.Text = $"Ход игрока: {currentPlayer}";

}