МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «КубГУ»)

Факультет компьютерных технологий и прикладной математики  
Кафедра информационных технологий

Отчет по индивидуальному заданию №1

**«Создание приложения “Крестики-Нолики”»**

Работу выполнил  
студент 42 группы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Б.М.Ибрагимов

Направление подготовки 01.03.02 Прикладная математика и информатика

Проверил  
канд. техн. наук, доц.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.А.Полупанов

Краснодар 2020

**Цель работы:** разработать игру с компьютером «Крестики-Нолики» на C#.

**Задача 1.** Разработать игру «Крестики-Нолики», с интерфейсом «человек-компьютер». Начальными параметрами будут:

1. Размеры игрового поля. Минимальное значение = 3\*3, Максимальное значение = 10\*10.
2. Количество клеток поля подряд заполненным одним символом, который будет считаться выигрышем. Минимальное значение = 3\*3, Максимальный = размер игрового поля.
3. Символ игрока. У компьютера автоматически другой противоположный символ.

**ЛИСТИНГ ПРОГРАММЫ**

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace CrossZero\_Games{

public partial class Form1 : Form{

string USER\_CHAR, AI\_CHAR, EMPTY\_CHAR = "\_";

byte sizePole, winerSize;

bool AI\_TURN = true, USER\_TURN = false;

Button[,] pole2;

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

public Form1(){

InitializeComponent();

}

private bool GetWiner2(Button[,] copyPole, int x, int y) {

bool checkWays;

string CHECK\_CHAR = copyPole[x, y].Text;

//------------------Right--------------------

checkWays = true;

if (y + winerSize <= sizePole) {

for (int k = y + 1; k < y + winerSize; k++) {

if (copyPole[x, k].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

}

if (checkWays) return true;

}

//------------------Left--------------------

checkWays = true;

if (y + 1 - winerSize >= 0) {

for (int k = y - 1; k > y - winerSize; k--) {

if (copyPole[x, k].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

}

if (checkWays) return true;

}

//------------------Down--------------------

checkWays = true;

if (x + winerSize <= sizePole) {

for (int k = x + 1; k < x + winerSize; k++) {

if (copyPole[k, y].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

}

if (checkWays) return true;

}

//------------------Up--------------------

checkWays = true;

if (x + 1 - winerSize >= 0) {

for (int k = x - 1; k > x - winerSize; k--) {

if (copyPole[k, y].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

}

if (checkWays) return true;

}

//------------------Right and Up--------------------

checkWays = true;

if (y + winerSize <= sizePole && x + 1 - winerSize >= 0) {

byte k1 = 1;

while (k1 < winerSize) {

if (copyPole[x - k1, y + k1].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

k1++;

}

if (checkWays) return true;

}

//------------------Left and Up--------------------

checkWays = true;

if (y + 1 - winerSize >= 0 && x + 1 - winerSize >= 0) {

byte k1 = 1;

while (k1 < winerSize) {

if (copyPole[x - k1, y - k1].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

k1++;

}

if (checkWays) return true;

}

//------------------Left and Down--------------------

checkWays = true;

if (y + 1 - winerSize >= 0 && x + winerSize <= sizePole) {

byte k1 = 1;

while (k1 < winerSize) {

if (copyPole[x + k1, y - k1].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

k1++;

}

if (checkWays) return true;

}

//------------------Right and Down--------------------

checkWays = true;

if (y + winerSize <= sizePole && x + winerSize <= sizePole) {

byte k1 = 1;

while (k1 < winerSize) {

if (copyPole[x + k1, y + k1].Text != CHECK\_CHAR) {

checkWays = false;

break;

}

k1++;

}

if (checkWays) return true;

}

return false;

}

private string GetMoveAI(){

string move = null;

List<string> possibleMoves = new List<string>();

for (int i = 0; i < sizePole; i++)

for (int j = 0; j < sizePole; j++)

if (pole2[i, j].Text == EMPTY\_CHAR) possibleMoves.Add(pole2[i, j].Name);

Random random = new Random();

return possibleMoves[random.Next(0,possibleMoves.Count)];

}

private void buttonPoleClick(object sender, EventArgs e){

sender.GetType().GetProperty("Text").SetValue(sender,USER\_CHAR);

sender.GetType().GetProperty("Enabled").SetValue(sender, false);

string s = sender.GetType().GetProperty("Name").GetValue(sender).ToString();

bool is\_user\_winer = GetWiner2(pole2, System.Convert.ToInt32(s.Split(' ')[0]), System.Convert.ToInt32(s.Split(' ')[1]));

if (is\_user\_winer){

this.label4.Text = "User is winer";

MessageBox.Show("User is winer");

for (int i = 0; i < sizePole; i++)

for (int j = 0; j < sizePole; j++)

pole2[i, j].Visible = false;

}

else{

for (int i = 0; i < sizePole; i++)

for (int j = 0; j < sizePole; j++)

if (pole2[i, j].Enabled) pole2[i, j].Visible = false;

string AIMove = GetMoveAI();

for (int i = 0; i < sizePole; i++){

for (int j = 0; j < sizePole; j++){

if (pole2[i, j].Text == EMPTY\_CHAR && pole2[i, j].Name == AIMove) {

pole2[i, j].Text = AI\_CHAR;

pole2[i, j].Enabled = false;

}

pole2[i, j].Visible = true;

}

}

bool winer = GetWiner2(pole2,System.Convert.ToInt32(AIMove.Split(' ')[0]), System.Convert.ToInt32(AIMove.Split(' ')[1]));

if (winer){

this.label4.Text = "AI is winer";

MessageBox.Show("AI is winer");

for (int i = 0; i < sizePole; i++)

for (int j = 0; j < sizePole; j++)

if (pole2[i, j].Enabled) pole2[i, j].Enabled = false;

}

}

}

private void button1\_Click(object sender, EventArgs e){

if (radioButton1.Checked || radioButton2.Checked){

if (this.radioButton1.Checked){

USER\_CHAR = radioButton1.Text;

AI\_CHAR = radioButton2.Text;

}

else{

USER\_CHAR = radioButton2.Text;

AI\_CHAR = radioButton1.Text;

}

scores.Add(USER\_CHAR, -100);

scores.Add(AI\_CHAR, 100);

scores.Add("DRAW", 0);

sizePole = System.Convert.ToByte(this.numericUpDown1.Value);

winerSize = System.Convert.ToByte(this.numericUpDown2.Value);

int heightButtons = this.groupBox2.Height / sizePole - 3;

Size buttonSize = new Size(this.groupBox2.Width / sizePole - 3, heightButtons);

Font buttonsFont = new Font("Arial", heightButtons / 2);

pole2 = new Button[sizePole, sizePole];

this.groupBox2.Visible = false;

for (int i = 0; i < sizePole; i++){

for (int j = 0; j < sizePole; j++){

Button button = new Button();

button.Name = i + " " + j;

button.Font = buttonsFont;

button.Text = "\_";

button.Size = buttonSize;

button.Location = new Point(this.groupBox2.Location.X + j \* button.Size.Width, this.groupBox2.Location.Y + i \* button.Size.Height);

button.Click += buttonPoleClick;

pole2[i, j] = button;

this.groupBox2.Controls.Add(button);

}

}

this.button1.Enabled = false;

this.groupBox2.Visible = true;

}

else {

MessageBox.Show("Выберите символ, чтобы начать игру!!!");

}

}

private void numericUpDown2\_Click(object sender, EventArgs e){

this.numericUpDown2.Maximum = this.numericUpDown1.Value;

}

private void numericUpDown1\_Click(object sender, EventArgs e){

if (this.numericUpDown2.Value > this.numericUpDown1.Value){

this.numericUpDown2.Value = this.numericUpDown1.Value;

}

}

}

}

**РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ**

На рисунке 1 изображено игровое поле 3\*3 в начале:

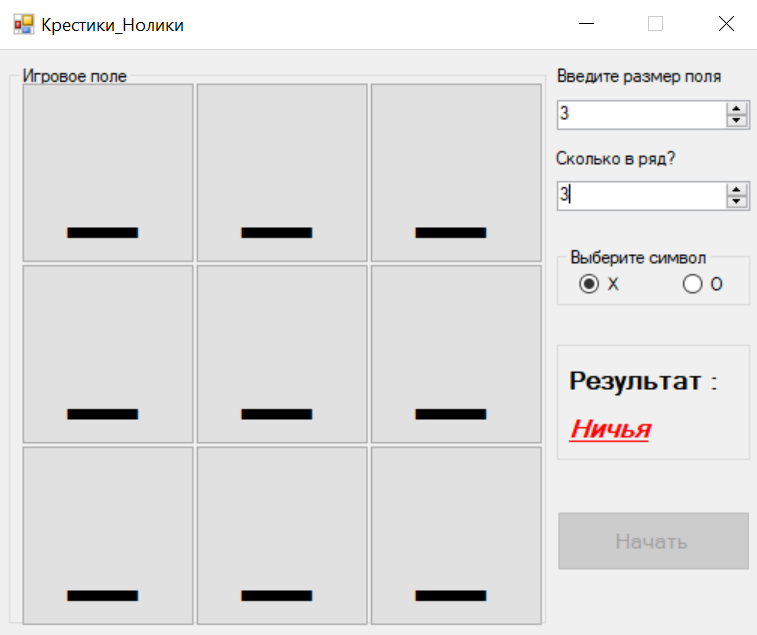


Рисунок 1 – игровое поле 3\*3 в начале игры

На рисунке 2 показано состояние игрового поля в процессе игры:

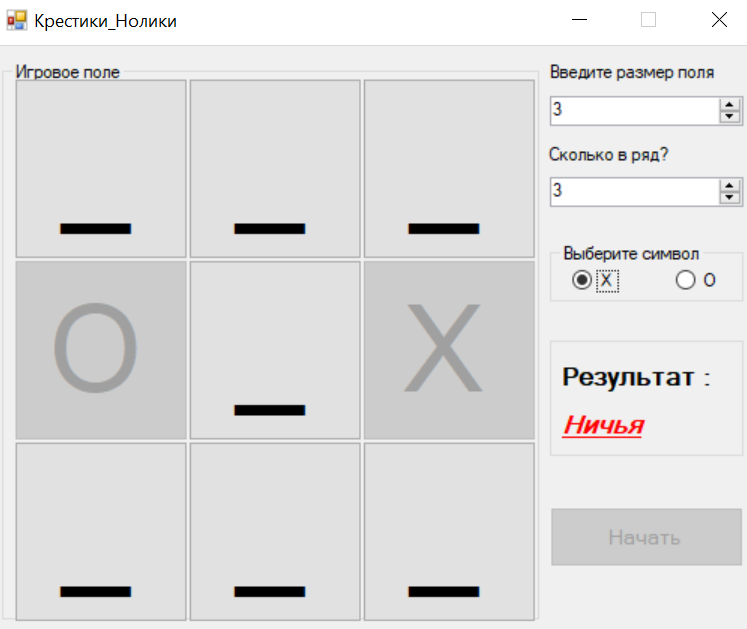


Рисунок 2 – игровое поле 3\*3 в процессе игры

При победе кого-либо выводится соответствующее сообщение.

**Вывод:** была написана игра «Крестики-Нолики» пользователя с компьютером, где в качестве начальных данных выступали размеры игрового поля, количество клеток поля, который может быть засчитан за победу, символ пользователя в данной игре. При победе кого-либо выводится соответствующее сообщение.