НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. Ігоря Сікорського»
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»

КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

КУРСОВА РОБОТА

з дисципліни

Програмування та алгоритмічні мови

на тему: Ігрова програма «П’ять в ряд» - хрестики нулики на «нескінченному» полі.

Студента 1 курсу групи КА-03
Галузь знань 124 Системний аналіз
Захаренко Нікіта Сергійович

Керівник _Старший викладач Назарчук Ірина Василівна

Національна оцінка _____
Кількість балів: _____Оцінка: ECTS _____

Члени комісії _____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)
_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали)
_____ _____
(підпис) (вчене звання, науковий ступінь, прізвище та ініціали

Київ - 2021рік

НТУУ „КПІ ім.І.Сікорського” ІПСА
(назва вищого закладу освіти)

Кафедра	***математичних методів системного аналізу***

Дисципліна	***Програмування та алгоритмічні мови***

Галузь знань	***12 Інформаційні технології***

Курс	***перший***	Група	***КА-03***	Семестр	***другий***

ЗАВДАННЯ
на курсовий проект(роботу) студента

Захаренко Нікіта Сергійович
(прізвище, ім’я, по батькові)

1. Тема проекту(роботи)	Ігрова програма «П’ять в ряд» - хрестики нулики на «нескінченному» полі.

2. Строк здачі студентом закінченого проекту(роботи)	***17.05.2021 р.***

3. Вихідні дані до проекту(роботи)
Гравці намагаються поставити в неперервний рядок п’ять хрестиків або нуликів. Гра реалізується між двома гравцями. Візуальний інтерфейс виконаний з використанням бібліотеки System::Drawing. Гравці ставлять хрестики або нолики за допомогою миші кліком на поле.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
1. Постановка задачі.
2. Метод розв’язку задачі
3. Загальна блок-схема алгоритму та опис алгоритму
4. Опис програмного продукту.
5. Результати роботи.
6. Висновки.
7. Список літератури.
Додаток А. Текст програми.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
1. Загальна блок-схема алгоритму.
2. Ілюстрації роботи програми.

6. Дата видачі завдання	16.02.2020

КАЛЕНДАРНИЙ ПЛАН

№/п	Назва етапів курсового проекту (роботи)	Строк виконання етапів роботи	Примітка
1.	Вибір теми курсової роботи. Опрацювання відповідної літератури. Оформлення листа Завдання.	20.02.2021	
2.	Аналіз постановки задачі.	27.02.2021	
3.	Вибір та дослідження методів, вибір відповідних структур даних, розробка алгоритму. Перше узгодження з керівником.	05.03.2021	
4.	Проектування інтерфейсу.	12.03.2021	
5.	Друге узгодження з керівником.	19.03.2021	
6.	Програмна реалізація.	26.03.2021	
7.	Демонстрація першого варіанту. Третє узгодження з керівником.	02.04.2021	
8.	Заключне тестування програми.	16.04.2021	
9.	Аналіз результатів. Оформлення звіту.	23.04.2021	
10.	Захист та демонстрація курсової роботи.	17.05.2021 – «A» 21.05.2021 –«B,C» 30.05.2021 – «D,E»	

Студент		
	(підпис)	

Керівник		
	(підпис)	(прізвище, ім’я, по батькові)

(дата)	

ЗМІСТ

Анотація

В курсовій роботі відтворена знайома ще з дитинства гра «хрестики-нолики», але з модифікаціями: для перемоги потрібно поставити 5 фігур в ряд та поле є нескінченним. Інша назва: «Гомоку». Програма виконана з використанням мови програмування C++ та бібліотеки “Microsoft Visual C++”, що дозволила зробити інтерфейс, чутливий до кліків миші.

Продукт курсової роботи має розважальний характер, гра відбувається між двома гравцями, кожен з гравців ходить по черзі.

Summary

In the course work the game "crosses-zeros" familiar from the childhood is reproduced, but with modifications: for a victory it is necessary to put 5 figures in a row and the field is infinite. Another name: "Homoku". The program is made using the C ++ programming language and the library "Microsoft Visual C ++", which allowed to make the interface sensitive to mouse clicks.

The product of the course work is entertaining, the game takes place between two players, each of the players takes turns.

ВСТУП

Актуальність: відома гра для релаксу «крестики-нолики» становиться лише цікавішою, якщо додати туди нескінченне поле та перемагати якщо маєш 5 фігур в ряд. Як і будь-яка гра, цей продукт є актуальним, наприклад, коли друг прийшов у гості.

Метою даної курсової роботи було реалізувати алгоритм
гри в «Гомоку» для двох гравців, зробити приємний оку інтерфейс.

Завдання: На основі власних знань та аналізу літературних джерел була виконана курсова робота. Основні етапи:

- Розробка гри.
- Дослідження та втілення інтерфейсу, що використовує Windows Forms.
- Виконання основного дизайну форми.

Були втілені та використані принципи роботи з Forms, що дозволило зробити управління в грі мишкою.

Практичне значення одержаних результатів: даний продукт може бути використаний для розважання удвох.

Використане програмне забезпечення: При написанні курсової роботи було використане середовище розробки Microsoft Visual Studio та операційна система Windows 10, для підготовки та оформлення курсової роботи використано Microsoft Word.

Структура роботи: Робота складається із вступу, трьох розділів, висновків, додатків та списку використаних джерел.

РОЗДІЛ 1
Постановка задачі

1.1. Огляд існуючих підходів до розв'язання поставленої задачі

Для розв'язання подібної задачі можливо було використати декілька способів. В основному це стосується реалізації нескінченного поля. Є два відомих мені путі розв'язання проблеми :

1) Використання «псевдонескінченного» поля, яке має великі розміри, а гравці переміщають «камеру по ньому».
2) Використання структури vector для того, щоб кожен раз, коли потрібно змістити поле, виділяти нову пам'ять під клітинки, тим самим роблячи поле нескінченним в, звісно, межах пам'яті комп'ютера.

Мета гри – поставити п'ять своїх фігур в ряд та не дозволити зробити цього супротивнику.

1.2. Уточнена постановка задачі на розробку програмного забезпечення

Постановка задачі полягає у створенні розважальної програми, яка буде повністю відтворювати алгоритм гри «хрестики-нолики» 5 в ряд на нескінченному полі, в якій:

- Поле нескінченне або псевдонескінченне
- Гравці ходять по черзі
- Для перемоги потрібно, щоб 5 однакових фігур(хрестиків або ноликів), стояли в рядок по горизонталі, вертикалі або діагоналі.
- По полю можна переміщатися, та якщо фігури виходять з зони їх «видимості», вони залишаються в пам'яті.

Отже, ці пункти мають бути виконані за допомогою мови програмування C++, мати приємний оку інтерфейс, для керування програмою користувач використовує мишку.

РОЗДІЛ 2

Розробка програмного продукту

2.1. Метод розв'язку задачі

Для написання даної програми була використана мова *C++* та бібліотека Microsoft Visual C++.

Для створення нескінченного поля мною був обраний варіант із використанням динамічного масиву великого розміру, по яку гравці можуть переміщати «камеру», яка має розміри 10x10. Для того, щоб поставити крестик або нулик та занести його до масиву, використовується такий принцип: береться координата кліку по об'єкту pictureBox1, після чого з'ясовується за допомогою циклу, в яку ячейку попадає координата кліку, після чого до лічильнику додаються координати «камери» в масиві та записується в великий двовимірний масив. Усі елементи на об'єкті pictureBox1, виконані за допомогою бібліотеки System::Drawing::Drawing2D.

Для детектингу перемоги використаний такий алгоритм: розглядається кожна непуста клітинка, яка знаходиться не ближче чим на 4 клітинки до границь масиву та перевіряються наступні після неї 4 ячейки, якщо всі вони рівні тому ж, що й в перевіряємій клітинці – перемога знайдена, та в залежності від того, чому дорівнює клітинка, повертається значення: 0, якщо перемога не знайдена, 1 якщо перемогли хрестики, 2 якщо перемогли нолики.

Для переміщення «камери» по полю було введено декілька функцій та змінних: змінні, що вказують на координати верхньої лівої клітинки з видимого поля на всьому полі, та функції переміщення камери в усі боки, та функція, яка відтворює на екрані масив в вигляді крестиків та нуликів.

Спочатку потрібно ввести розміри поля, для якого потрібно виділити пам'ять. Логічно, що чим більше розміри поля – тим складніше комп'ютеру обробляти інформацію, але тим більше простору для гри з'являється. За умовчанням стоїть поле 30x30. Після чого потрібно натиснути кнопку «Очистити поле» і можна починати грати! Для гри потрібно просто

натискати на вільні ячейки поля, якщо ячейка зайнята – нічого не відбудеться.

2.2. Алгоритм розв’язку задачі

Структурна блок-схема алгоритму роботи програми зображена на рисунку 2.1.

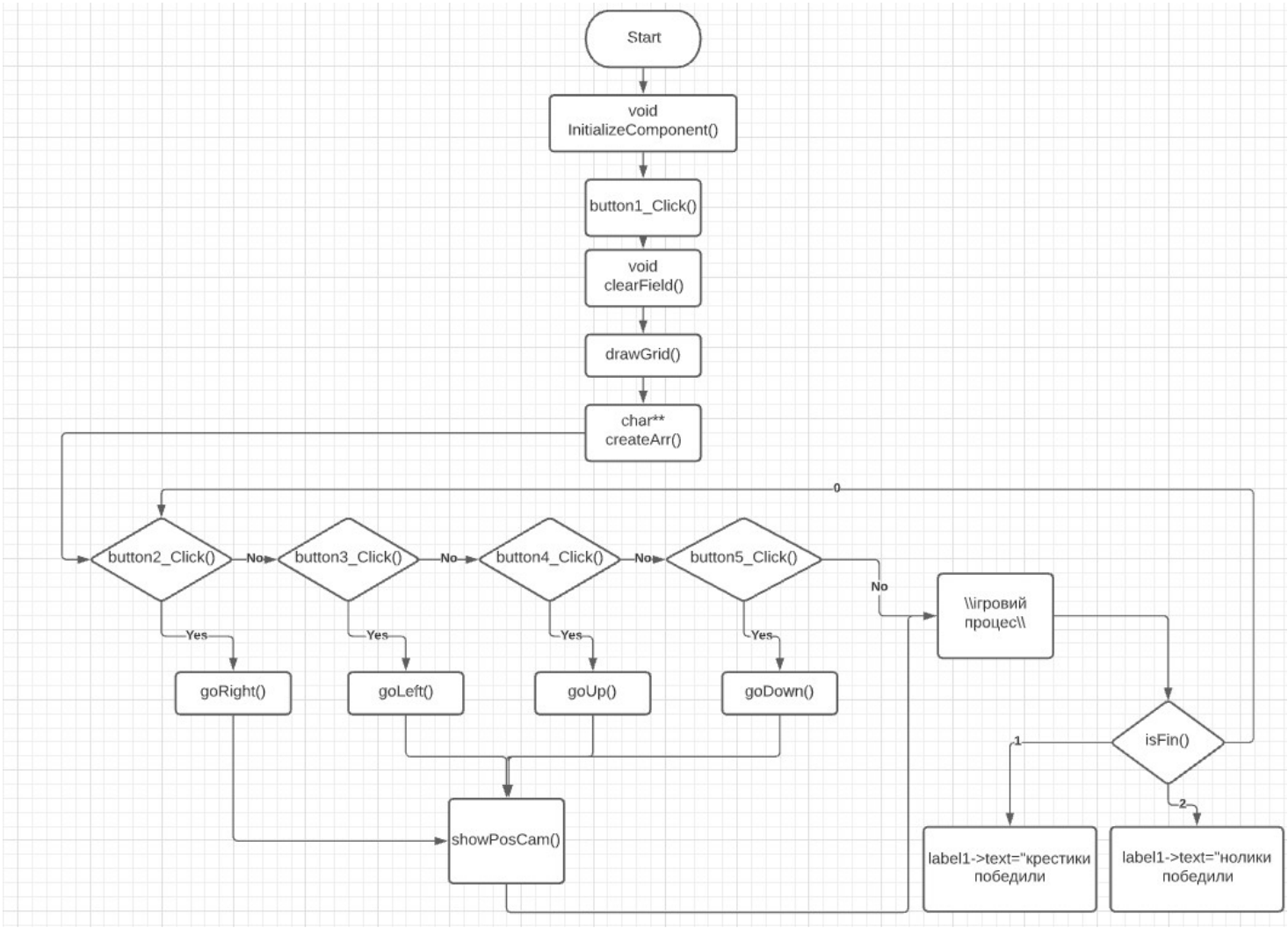


Рис. 2.1. Структурна блок-схема алгоритму роботи програми

Програма очікує натискання кнопки «Очистити поле», після чого очікує або натискання на кнопки переміщення камери, або натискання на pictureBox1, яке фігурує під виглядом \\ігровий процес\\.

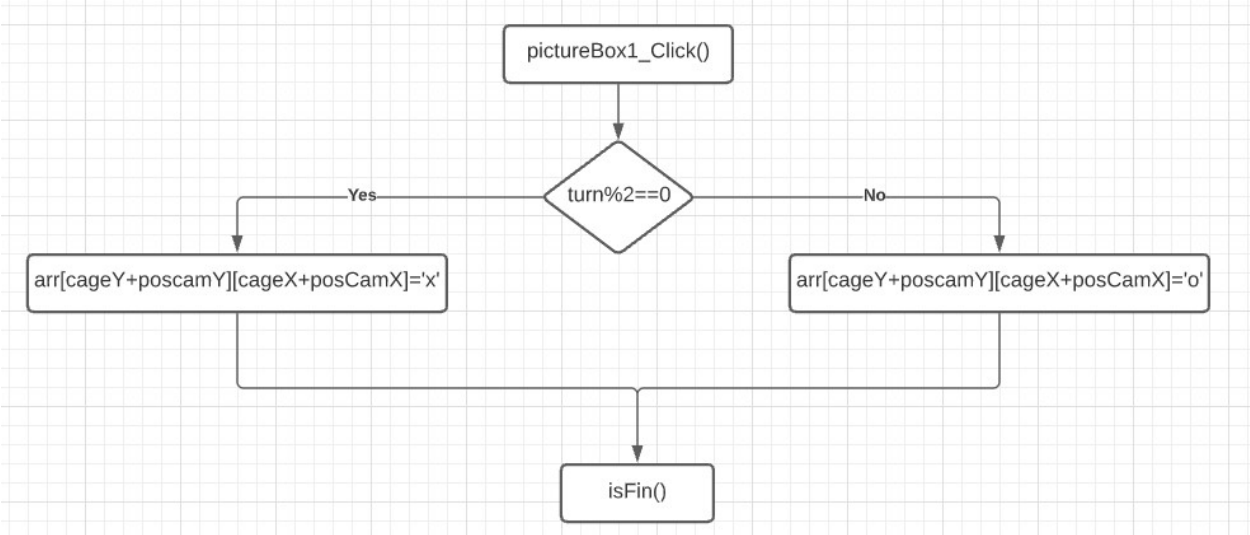


Рис. 2.2. Структурна блок-схема алгоритму роботи ігрового процесу програми.

РОЗДІЛ 3

Опис розробленого програмного продукту

3.1. Опис головних структур і змінних програми

Опис головних структур і змінних програми наведено в таблиці 3.1.

Таблиця 3.1. Опис головних змінних програми

№	Змінна	Тип змінної	Призначення змінної
1	InitializeComponent	void	Ініціалізація усіх компонентів на формі. Функція, відповідальна за дизайн форми.
2	turn	Int	Змінна стану ходу.
3	g	Graphics	Об’єкт класу Graphics, відповідальний за малювання на полі.
4	arr	char**	Масив, який відповідає клітинкам на полі.
5	sizeX, sizeY	int	Розміри масиву arr.
6	scale	Int	Кількість видимих клітинок.
7	poxCamX, posCamY	Int	Координати «камери»

3.2 Опис головних функцій програми

Опис головних функцій програми наведено в таблиці 3.2.

Таблиця 3.2. Опис головних функцій програми

№	Синтаксис	Опис
1	Char**createArr(int m, int n)	Функція створення динамічного масиву.
2	Int isFin()	Функція перевірки на перемогу.
3	Void drawCross(float x, float y) void drawCircle(float x, float y)	Малює хрестик або нулик по заданих координатах
4	voiddrawField()	Малює видиму частину масиву на полі в вигляді крестиків та ноликів.
5	Void drawGrid()	Малює сітку для поля.
6	Void goRight() Void goLeft() Void goUp() void goDown()	Функції переміщення камери по масиву.
7	Void clearField()	Очищує поле та масив.

3.3. Опис інтерфейсу

На початку роботи, користувач попадає одразу ж до гри. Йому потрібно лише налаштувати якого розміру виділити масив та скільки клітинок буде видно на екрані та натиснути «Очистити поле» (Рис. 3.1).

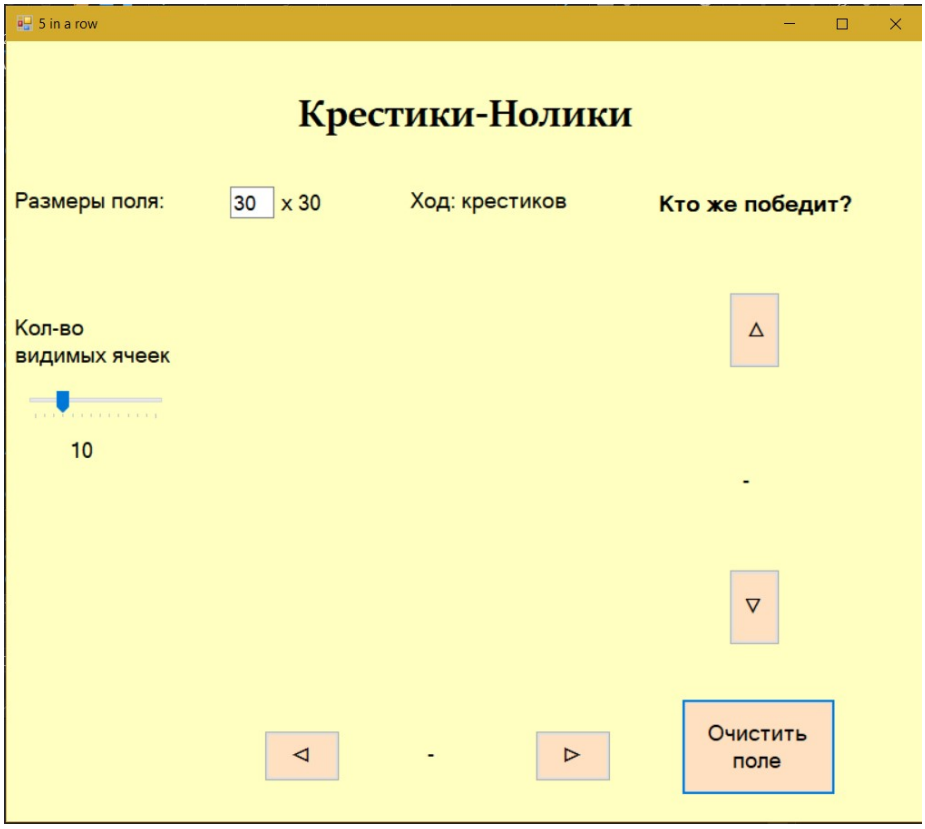


Рис. 3.1 Початковий вигляд програми

Після натиснення кнопки «Очистить поле», користувачить бачить поле, на якому вже може розпочинати гру(рис. 3.2).

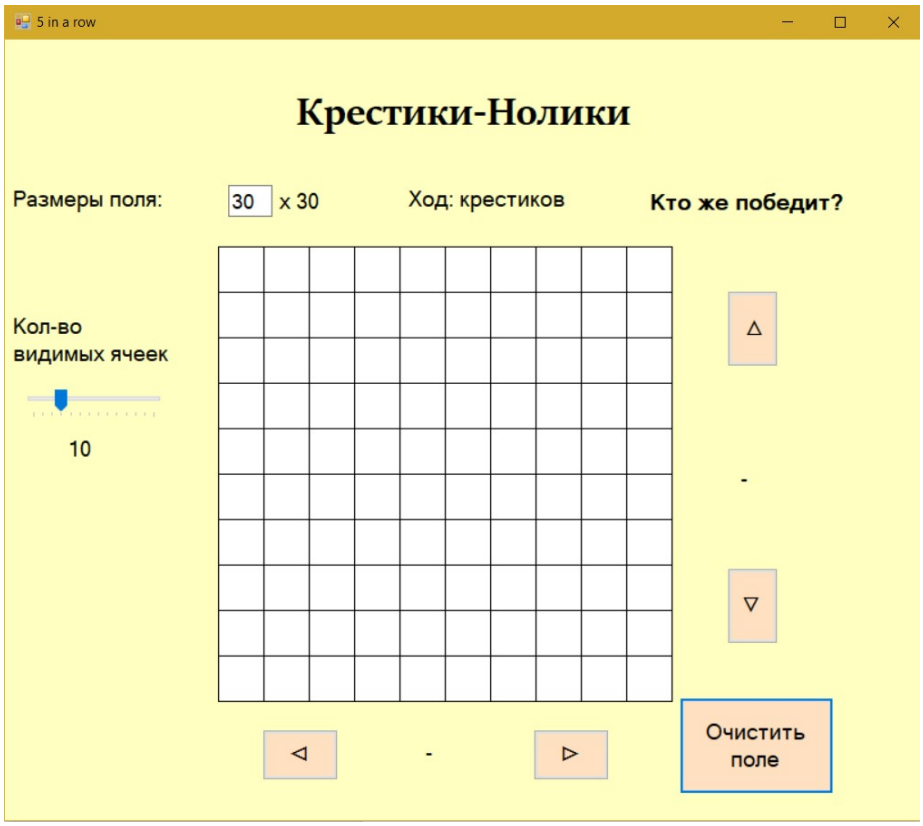


Рис. 3.2 Вигляд програми після натискання «Очистить поле»

На рис 3.3 показано як виглядає ігровий процес програми.

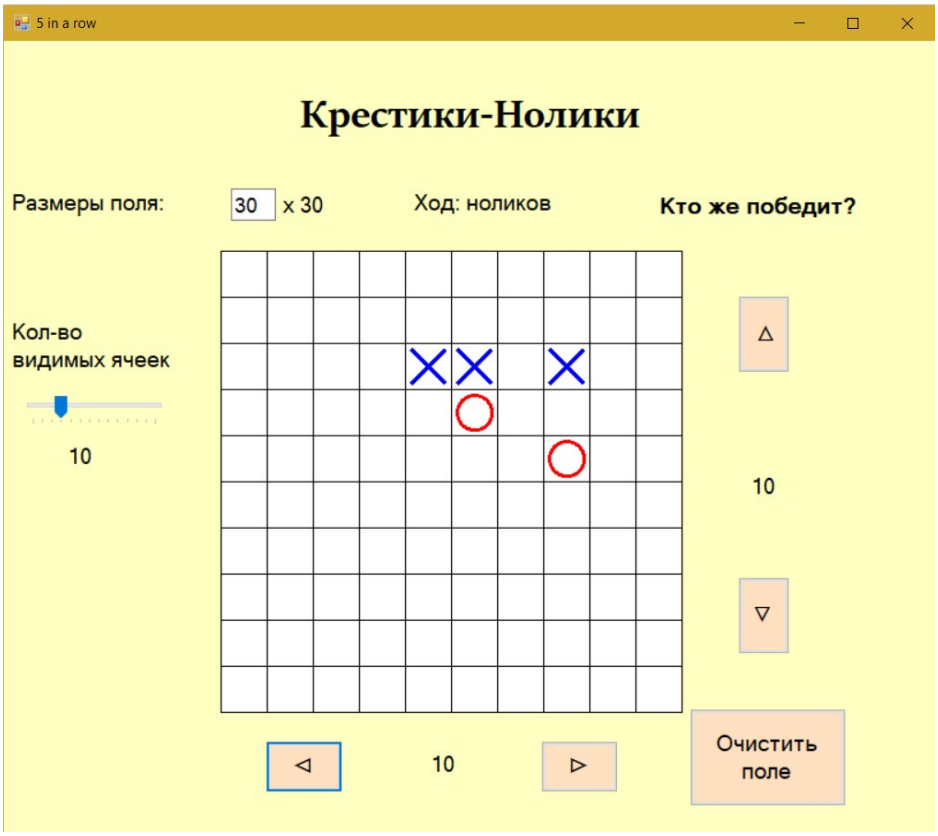
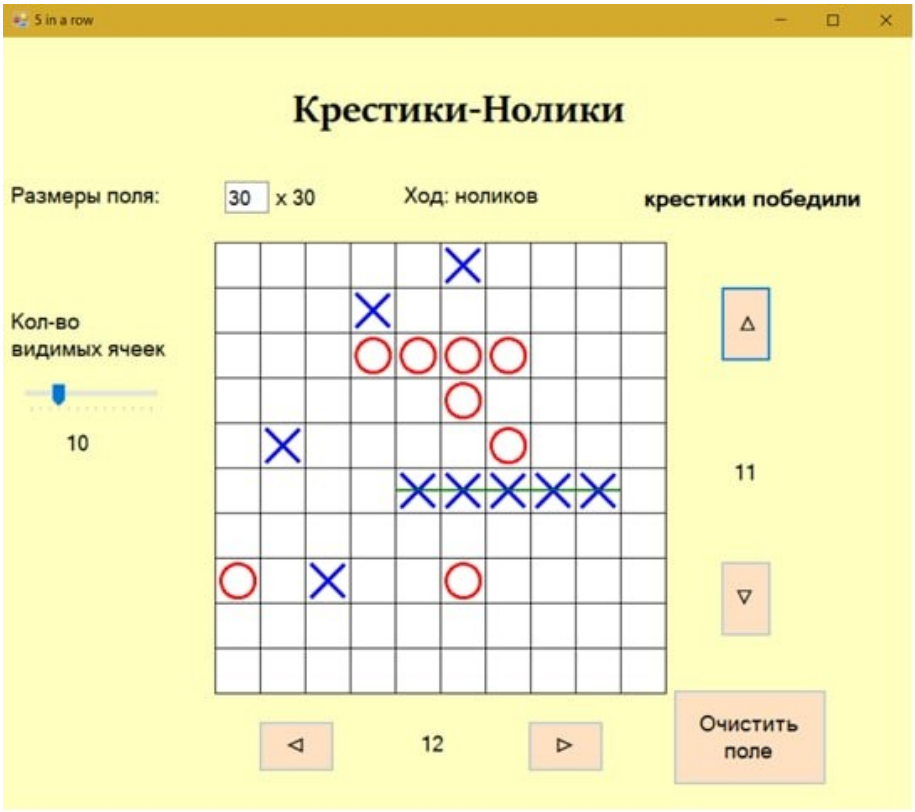


Рис. 3.3 Вигляд програми в процесі гри



Так виглядає перемога одного з гравців(рис 3.4)

Рис. 3.4 Вигляд програми коли один з гравців переміг

На рисунку 3.5 показано, як виглядатиме гра, якщо поміняти налаштування кількості видимих клітинок.

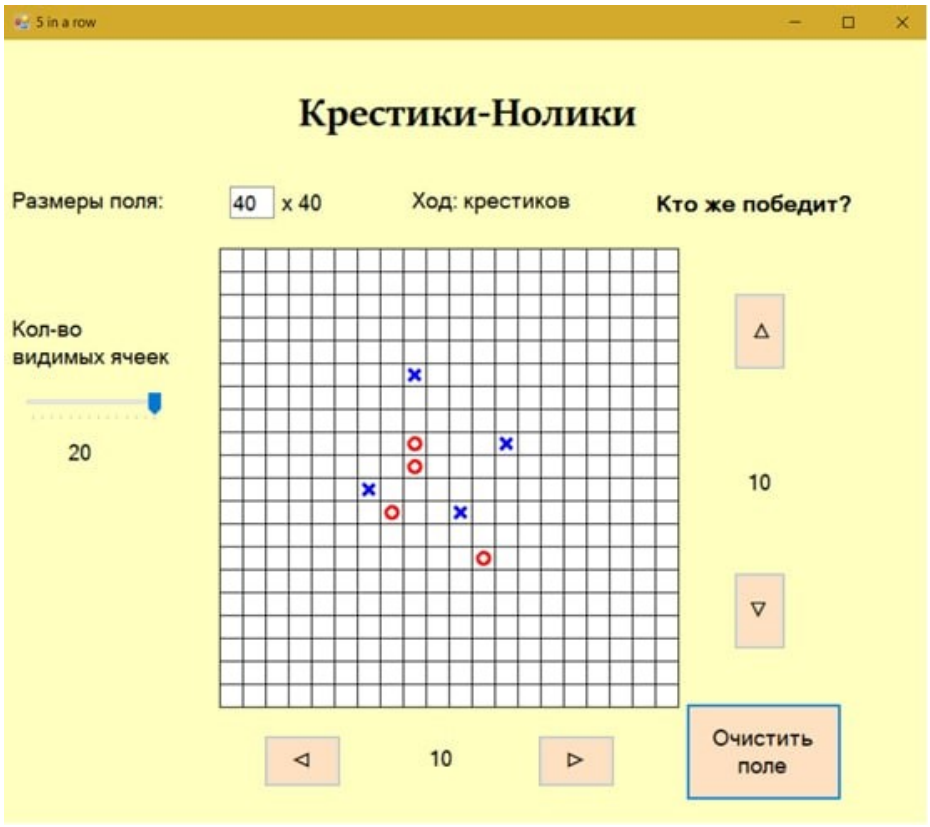


Рис. 3.5 Вигляд гри з іншими параметрами кастомізації

ВИСНОВКИ

Під час розробки даного програмного продукту, я навчився використовувати мову C++ для розробки десктопних аплікацій з GUI, що викроистовує для комунікації з користувачем не клавіатуру, а мишу. Також навчився працювати із графікою в бібліотеці System::Drawing::Drawing2D.

За допомогою мови програмування C++ вдалося реалізувати сприятливий графічний інтерфейс. Основним середовищем розробки було Microsoft Visual Studio.

Перевагами даної роботи з-понад аналогів, я вважаю досить велику можливість кастомізації грального поля.

Основним недоліком даної програми автор вважає обмеженність грального поля в якихось конкретних рамках.

Можливе подальше покращення в вигляді створення штучного інтелекту для гри проти нього.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Bjarne Stroustrup Programming: principles and practice using C++ – Second edition, 2014 – 2339c.

2. Шилдт, Герберт. C++: руководство для начинающих, 2-е издание: Пер. с англ. – М.: Издательский дом «Вильямс», 2005 – 672с.

3. Програмування та алгоритмічні мови. Програмування: Методичні вказівки до виконання курсового проекту для студентів галузі знань 12 «Інформаційні технології» спеціальність 122 «Комп'ютерні науки та інформаційні технології», 124 «Системний аналіз» / Уклад.: І.В.Назарчук, Г.Г.Швачко – К.НТУУ «КПІ», 2017 – 34с.

Додаток А

Текст програми

```
#pragma once
#include <string.h>
#include <string>
using namespace std;
namespace CppCLRWinformsProjekt {
	using namespace System;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Windows::Forms;
	using namespace System::Data;
	using namespace System::Drawing;

	/// <summary>
	/// Zusammenfassung fьr Form1
	/// </summary>
	///

	public ref class Form1 : public System::Windows::Forms::Form
	{
	public:
		Form1(void)
		{
			InitializeComponent();
			//
			//TODO: Konstruktorcode hier hinzufьgen.
			//

		}

	protected:
		/// <summary>
		/// Verwendete Ressourcen bereinigen.
		/// </summary>
		~Form1()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::PictureBox^ pictureBox1;
	private: System::Windows::Forms::Button^ button1;
	private: System::Windows::Forms::DataGridView^ dataGridView1;
	private: System::Windows::Forms::Label^ label1;
	private: System::Windows::Forms::Button^ button2;
	private: System::Windows::Forms::Button^ button3;
	private: System::Windows::Forms::Button^ button4;
	private: System::Windows::Forms::Button^ button5;
	private: System::Windows::Forms::Label^ label2;
```

```
        private: System::Windows::Forms::Label^ label3;




        private: System::Windows::Forms::TextBox^ textBox1;

        private: System::Windows::Forms::Label^ label6;
        private: System::Windows::Forms::Label^ label7;
        private: System::Windows::Forms::Label^ label4;

        private: System::Windows::Forms::Label^ label8;
        private: System::Windows::Forms::TrackBar^ trackBar1;
        private: System::Windows::Forms::Label^ label5;
        private: System::Windows::Forms::Label^ label9;




        protected:

        private:
                /// <summary>
                /// Erforderliche Designervariable.
                /// </summary>
                System::ComponentModel::Container^ components;

#pragma region Windows Form Designer generated code
                /// <summary>
                /// Erforderliche Methode fьr die Designerunterstьtzung.
                /// Der Inhalt der Methode darf nicht mit dem Code-Editor geдndert werden.
                /// </summary>
                void InitializeComponent(void)
                {
                        this->pictureBox1 = (gcnew System::Windows::Forms::PictureBox());
                        this->button1 = (gcnew System::Windows::Forms::Button());
                        this->dataGridView1 = (gcnew
System::Windows::Forms::DataGridView());
                        this->label1 = (gcnew System::Windows::Forms::Label());
                        this->button2 = (gcnew System::Windows::Forms::Button());
                        this->button3 = (gcnew System::Windows::Forms::Button());
                        this->button4 = (gcnew System::Windows::Forms::Button());
                        this->button5 = (gcnew System::Windows::Forms::Button());
                        this->label2 = (gcnew System::Windows::Forms::Label());
                        this->label3 = (gcnew System::Windows::Forms::Label());
                        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                        this->label6 = (gcnew System::Windows::Forms::Label());
                        this->label7 = (gcnew System::Windows::Forms::Label());
                        this->label4 = (gcnew System::Windows::Forms::Label());
                        this->label8 = (gcnew System::Windows::Forms::Label());
                        this->trackBar1 = (gcnew System::Windows::Forms::TrackBar());
                        this->label5 = (gcnew System::Windows::Forms::Label());
```

```
            this->label9 = (gcnew System::Windows::Forms::Label());
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->pictureBox1))->BeginInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->dataGridView1))->BeginInit();
            (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->trackBar1))->BeginInit();
            this->SuspendLayout();
            //
            // pictureBox1
            //
            this->pictureBox1->Enabled = false;
            this->pictureBox1->Location = System::Drawing::Point(188, 182);
            this->pictureBox1->Name = L"pictureBox1";
            this->pictureBox1->Size = System::Drawing::Size(401, 401);
            this->pictureBox1->TabIndex = 0;
            this->pictureBox1->TabStop = false;
            this->pictureBox1->LoadCompleted += gcnew System::ComponentModel::AsyncCompletedEventHandler(this, &Form1::pictureBox1_LoadCompleted);
            this->pictureBox1->SizeChanged += gcnew System::EventHandler(this, &Form1::pictureBox1_SizeChanged);
            this->pictureBox1->MouseClick += gcnew System::Windows::Forms::MouseEventHandler(this, &Form1::pictureBox1_MouseClick);
            //
            // button1
            //
            this->button1->BackColor = System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255)), static_cast<System::Int32>(static_cast<System::Byte>(224)),
                    static_cast<System::Int32>(static_cast<System::Byte>(192)));
            this->button1->Location = System::Drawing::Point(595, 579);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(136, 85);
            this->button1->TabIndex = 1;
            this->button1->Text = L"Очистить поле";
            this->button1->UseVisualStyleBackColor = false;
            this->button1->TextChanged += gcnew System::EventHandler(this, &Form1::button1_TextChanged);
            this->button1->Click += gcnew System::EventHandler(this, &Form1::button1_Click);
            //
            // dataGridView1
            //
            this->dataGridView1->AutoSizeColumnsMode = System::Windows::Forms::DataGridViewAutoSizeColumnsMode::AllCells;
            this->dataGridView1->AutoSizeRowsMode = System::Windows::Forms::DataGridViewAutoSizeRowsMode::AllCells;
            this->dataGridView1->ColumnHeadersHeightSizeMode = System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
            this->dataGridView1->Location = System::Drawing::Point(197, 182);
            this->dataGridView1->Name = L"dataGridView1";
            this->dataGridView1->RowHeadersWidth = 51;
```

```
            this->dataGridView1->RowTemplate->Height = 24;
            this->dataGridView1->Size = System::Drawing::Size(401, 401);
            this->dataGridView1->TabIndex = 2;
            this->dataGridView1->Visible = false;
            //
            // label1
            //
            this->label1->AutoSize = true;
            this->label1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans
Serif", 13.8F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label1->Location = System::Drawing::Point(565, 131);
            this->label1->Name = L"label1";
            this->label1->Size = System::Drawing::Size(231, 29);
            this->label1->TabIndex = 3;
            this->label1->Text = L" Кто же победит\?";
            this->label1->Click += gcnew System::EventHandler(this,
&Form1::label1_Click);
            //
            // button2
            //
            this->button2->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255
)), static_cast<System::Int32>(static_cast<System::Byte>(224)),
                static_cast<System::Int32>(static_cast<System::Byte>(192)));
            this->button2->Location = System::Drawing::Point(466, 607);
            this->button2->Name = L"button2";
            this->button2->Size = System::Drawing::Size(67, 45);
            this->button2->TabIndex = 4;
            this->button2->Text = L"▷";
            this->button2->UseVisualStyleBackColor = false;
            this->button2->Click += gcnew System::EventHandler(this,
&Form1::button2_Click);
            //
            // button3
            //
            this->button3->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255
)), static_cast<System::Int32>(static_cast<System::Byte>(224)),
                static_cast<System::Int32>(static_cast<System::Byte>(192)));
            this->button3->Location = System::Drawing::Point(227, 607);
            this->button3->Name = L"button3";
            this->button3->Size = System::Drawing::Size(67, 45);
            this->button3->TabIndex = 5;
            this->button3->Text = L"◁";
            this->button3->UseVisualStyleBackColor = false;
            this->button3->Click += gcnew System::EventHandler(this,
&Form1::button3_Click);
            //
            // button4
            //
```

```
                this->button4->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255
)), static_cast<System::Int32>(static_cast<System::Byte>(224)),
                        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                this->button4->Location = System::Drawing::Point(637, 221);
                this->button4->Name = L"button4";
                this->button4->Size = System::Drawing::Size(45, 67);
                this->button4->TabIndex = 6;
                this->button4->Text = L" Δ";
                this->button4->UseVisualStyleBackColor = false;
                this->button4->Click += gcnew System::EventHandler(this,
&Form1::button4_Click);
                //
                // button5
                //
                this->button5->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255
)), static_cast<System::Int32>(static_cast<System::Byte>(224)),
                        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                this->button5->Location = System::Drawing::Point(637, 465);
                this->button5->Name = L"button5";
                this->button5->Size = System::Drawing::Size(45, 67);
                this->button5->TabIndex = 7;
                this->button5->Text = L"∇";
                this->button5->UseVisualStyleBackColor = false;
                this->button5->Click += gcnew System::EventHandler(this,
&Form1::button5_Click);
                //
                // label2
                //
                this->label2->AutoSize = true;
                this->label2->Location = System::Drawing::Point(367, 615);
                this->label2->Name = L"label2";
                this->label2->Size = System::Drawing::Size(21, 29);
                this->label2->TabIndex = 8;
                this->label2->Text = L"-";
                this->label2->Click += gcnew System::EventHandler(this,
&Form1::label2_Click);
                //
                // label3
                //
                this->label3->AutoSize = true;
                this->label3->Location = System::Drawing::Point(645, 374);
                this->label3->Name = L"label3";
                this->label3->Size = System::Drawing::Size(21, 29);
                this->label3->TabIndex = 9;
                this->label3->Text = L"-";
                //
                // textBox1
                //
                this->textBox1->Location = System::Drawing::Point(197, 128);
                this->textBox1->Name = L"textBox1";
                this->textBox1->Size = System::Drawing::Size(39, 34);
```

```
            this->textBox1->TabIndex = 15;
            this->textBox1->Text = L"30";
            this->textBox1->TextChanged += gcnew System::EventHandler(this,
&Form1::textBox1_TextChanged);
            //
            // label6
            //
            this->label6->AutoSize = true;
            this->label6->Location = System::Drawing::Point(3, 128);
            this->label6->Name = L"label6";
            this->label6->Size = System::Drawing::Size(188, 29);
            this->label6->TabIndex = 17;
            this->label6->Text = L"Размеры поля:";
            this->label6->Click += gcnew System::EventHandler(this,
&Form1::label6_Click);
            //
            // label7
            //
            this->label7->AutoSize = true;
            this->label7->Location = System::Drawing::Point(238, 130);
            this->label7->Name = L"label7";
            this->label7->Size = System::Drawing::Size(56, 29);
            this->label7->TabIndex = 18;
            this->label7->Text = L"x 30";
            //
            // label4
            //
            this->label4->AutoSize = true;
            this->label4->Font = (gcnew System::Drawing::Font(L"Sitka Banner",
28.2F, System::Drawing::FontStyle::Bold, System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(204)));
            this->label4->Location = System::Drawing::Point(247, 31);
            this->label4->Name = L"label4";
            this->label4->Size = System::Drawing::Size(403, 69);
            this->label4->TabIndex = 19;
            this->label4->Text = L"Крестики-Нолики";
            //
            // label8
            //
            this->label8->AutoSize = true;
            this->label8->Location = System::Drawing::Point(352, 128);
            this->label8->Name = L"label8";
            this->label8->Size = System::Drawing::Size(188, 29);
            this->label8->TabIndex = 21;
            this->label8->Text = L"Ход: крестиков";
            //
            // trackBar1
            //
            this->trackBar1->Location = System::Drawing::Point(12, 306);
            this->trackBar1->Maximum = 20;
            this->trackBar1->Minimum = 7;
            this->trackBar1->Name = L"trackBar1";
```

```
                this->trackBar1->RightToLeft =
System::Windows::Forms::RightToLeft::No;
                this->trackBar1->Size = System::Drawing::Size(133, 56);
                this->trackBar1->TabIndex = 22;
                this->trackBar1->Value = 10;
                this->trackBar1->Scroll += gcnew System::EventHandler(this,
&Form1::trackBar1_Scroll);
                //
                // label5
                //
                this->label5->AutoSize = true;
                this->label5->Location = System::Drawing::Point(3, 240);
                this->label5->Name = L"label5";
                this->label5->Size = System::Drawing::Size(190, 58);
                this->label5->TabIndex = 23;
                this->label5->Text = L"Кол-во \r\nвидимых ячеек";
                //
                // label9
                //
                this->label9->AutoSize = true;
                this->label9->Location = System::Drawing::Point(52, 348);
                this->label9->Name = L"label9";
                this->label9->Size = System::Drawing::Size(39, 29);
                this->label9->TabIndex = 24;
                this->label9->Text = L"10";
                //
                // Form1
                //
                this->AutoScaleDimensions = System::Drawing::SizeF(14, 29);
                this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
                this->BackColor =
System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<System::Byte>(255
)), static_cast<System::Int32>(static_cast<System::Byte>(255)),
                        static_cast<System::Int32>(static_cast<System::Byte>(192)));
                this->ClientSize = System::Drawing::Size(808, 687);
                this->Controls->Add(this->label9);
                this->Controls->Add(this->label5);
                this->Controls->Add(this->trackBar1);
                this->Controls->Add(this->label8);
                this->Controls->Add(this->label4);
                this->Controls->Add(this->label7);
                this->Controls->Add(this->label6);
                this->Controls->Add(this->textBox1);
                this->Controls->Add(this->label3);
                this->Controls->Add(this->label2);
                this->Controls->Add(this->button5);
                this->Controls->Add(this->button4);
                this->Controls->Add(this->button3);
                this->Controls->Add(this->button2);
                this->Controls->Add(this->label1);
                this->Controls->Add(this->dataGridView1);
                this->Controls->Add(this->button1);
                this->Controls->Add(this->pictureBox1);
```

```
                this->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif",
13.8F, System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->Margin = System::Windows::Forms::Padding(5);
                this->Name = L"Form1";
                this->Text = L"5 in a row";
                this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
                this->Click += gcnew System::EventHandler(this,
&Form1::Form1_Click);
                this->Paint += gcnew System::Windows::Forms::PaintEventHandler(this,
&Form1::Form1_Paint);
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>pictureBox1))->EndInit();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>dataGridView1))->EndInit();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this-
>trackBar1))->EndInit();
                this->ResumeLayout(false);
                this->PerformLayout();

        }
#pragma endregion
        int turn = 0;
        Graphics^ g;
        int sizeX = 30, sizeY = 30;
        char** arr;
        int scale = 10;
        int posCamX = sizeX / 2 - scale / 2;
        int posCamY = sizeY / 2 - scale / 2;

    private: System::Void Form1_Paint(System::Object^ sender,
System::Windows::Forms::PaintEventArgs^ e) {
    }
        char** createArr(int m, int n) {
            arr = new char* [m];
            for (int i = 0; i < m; i++) {
                arr[i] = new char[n];
            }
            for (int i = 0; i < m; i++)
            {
                for (int j = 0; j < n; j++)
                {
                    arr[i][j] = ' ';
                }
            }
            return arr;
        }
        int isFin()
        {
            g = pictureBox1->CreateGraphics();
            int width = pictureBox1->Width;
            int height = pictureBox1->Height;
            Pen^ greenPen = gcnew Pen(Color::Green, 2);
```

```
//vertCheck
for (int i = 0; i < sizeX; i++)
{

	for (int j = 0; j < sizeY - 4; j++)
	{
		if (arr[i][j] == ' ')
		{
			continue;
		}
		else if (arr[i][j] == 'x' && arr[i][j + 1] == 'x' && arr[i][j + 2] == 'x' && arr[i][j + 3] == 'x' && arr[i][j + 4] == 'x')
		{
			label1->Text = "Крестики победили";
			g->DrawLine(greenPen, (j-posCamX) * width / scale, (i-posCamY) * height / scale + height / (2 * scale), (j-posCamX + 5) * width / scale, (i-posCamY) * height / scale + height / (2 * scale));
			return 1;
		}
		else if (arr[i][j] == 'o' && arr[i][j + 1] == 'o' && arr[i][j + 2] == 'o' && arr[i][j + 3] == 'o' && arr[i][j + 4] == 'o')
		{
			label1->Text = "Нолики победили";
			g->DrawLine(greenPen, (j - posCamX) * width / scale, (i - posCamY) * height / scale + height / (2 * scale), (j - posCamX + 5) * width / scale, (i - posCamY) * height / scale + height / (2 * scale));
			return 2;
		}
	}
}
//horizCheck
for (int i = 0; i < sizeX - 4; i++)
{
	for (int j = 0; j < sizeY; j++)
	{
		if (arr[i][j] == 'x' && arr[i + 1][j] == 'x' && arr[i + 2][j] == 'x' && arr[i + 3][j] == 'x' && arr[i + 4][j] == 'x')
		{
			label1->Text = "Крестики победили";
			g->DrawLine(greenPen, (j - posCamX) * width / scale + width / (2 * scale), (i-posCamY) * height / scale, (j - posCamX) * width / scale + width / (2 * scale), (i - posCamY + 5) * height / scale);

			return 1;
		}
		if (arr[i][j] == 'o' && arr[i + 1][j] == 'o' && arr[i + 2][j] == 'o' && arr[i + 3][j] == 'o' && arr[i + 4][j] == 'o')
		{
			label1->Text = "Нолики победили";
			g->DrawLine(greenPen, (j - posCamX) * width / scale + width / (2 * scale), (i - posCamY) * height / scale, (j - posCamX) * width / scale + width / (2 * scale), (i - posCamY + 5) * height / scale);
			return 2;
```

```
                        }
                    }
                }
                //mainDiagCheck
                for (int i = 0; i < sizeX - 4; i++)
                {
                    for (int j = 0; j < sizeY - 4; j++)
                    {
                        if (arr[i][j] == 'x' && arr[i + 1][j + 1] == 'x' && arr[i + 2][j + 2] == 'x' && arr[i + 3][j + 3] == 'x' && arr[i + 4][j + 4] == 'x')
                        {
                            label1->Text = "Крестики победили";
                            g->DrawLine(greenPen, (j-posCamX) * width / scale, (i - posCamY) * height / scale, (j - posCamX + 5) * width / scale, (i - posCamY + 5) * height / scale);

                            return 1;
                        }
                        if (arr[i][j] == 'o' && arr[i + 1][j + 1] == 'o' && arr[i + 2][j + 2] == 'o' && arr[i + 3][j + 3] == 'o' && arr[i + 4][j + 4] == 'o')
                        {
                            label1->Text = "Нолики победили";
                            g->DrawLine(greenPen, (j - posCamX) * width / scale, (i - posCamY) * height / scale, (j - posCamX + 5) * width / scale, (i - posCamY + 5) * height / scale);
                            return 2;
                        }
                    }
                }
                //secondDiagCheck
                for (int i = 0; i < sizeY - 4; i++)
                {
                    for (int j = 4; j < sizeX - 1; j++)
                    {
                        if (arr[i][j] == 'x' && arr[i + 1][j - 1] == 'x' && arr[i + 2][j - 2] == 'x' && arr[i + 3][j - 3] == 'x' && arr[i + 4][j - 4] == 'x')
                        {
                            label1->Text = "Крестики победили";
                            g->DrawLine(greenPen, (j - posCamX + 1) * width / scale, (i - posCamY) * height / scale, (j - posCamX - 4) * width / scale, (i - posCamY + 5) * height / scale);

                            return 1;
                        }
                        if (arr[i][j] == 'o' && arr[i + 1][j - 1] == 'o' && arr[i + 2][j - 2] == 'o' && arr[i + 3][j - 3] == 'o' && arr[i + 4][j - 4] == 'o')
                        {
                            label1->Text = "Нолики победили";
                            g->DrawLine(greenPen, (j - posCamX + 1) * width / scale, (i - posCamY) * height / scale, (j - posCamX - 4) * width / scale, (i - posCamY + 5) * height / scale);
                            return 2;
                        }
```

```
                        }
                    }
                    return 0;
                }
                void drawCross(float x, float y)
                {
                    g = pictureBox1->CreateGraphics();
                    Pen^ bluePen = gcnew Pen(Color::Blue, 3);
                    g->DrawLine(bluePen, x + 5, y + 5, x - 5 + pictureBox1->Width / scale,
y - 5 + pictureBox1->Height / scale);
                    g->DrawLine(bluePen, x - 5 + pictureBox1->Width / scale, y + 5, x + 5,
y - 5 + pictureBox1->Height / scale);
                }
                void drawCircle(float x, float y)
                {
                    g = pictureBox1->CreateGraphics();
                    Pen^ redPen = gcnew Pen(Color::Red, 3);
                    g->DrawEllipse(redPen, x + 5, y + 5, (float)pictureBox1->Width / scale -
10, (float)pictureBox1->Height / scale - 10);
                }
        private: System::Void Form1_Load(System::Object^ sender, System::EventArgs^ e) {
                createArr(sizeX, sizeY);
                //dataGridView1->ColumnCount = sizeX;
                //dataGridView1->RowCount = sizeY;
        }
        private: System::Void pictureBox1_MouseClick(System::Object^ sender,
System::Windows::Forms::MouseEventArgs^ e) {
                int fin = isFin();

                float clickX = e->X;
                float clickY = e->Y;
                int cageY = -1, cageX = -1;
                for (int i = 0; i < pictureBox1->Width; i += pictureBox1->Width / scale,
cageX++)
                {
                    if (clickX<i && clickX>i - pictureBox1->Width / scale)
                    {
                        clickX = i - pictureBox1->Width / scale;
                        break;
                    }
                }
                for (int i = 0; i < pictureBox1->Height; i += pictureBox1->Height / scale,
cageY++)
                {
                    if (clickY<i && clickY>i - pictureBox1->Height / scale)
                    {
                        clickY = i - pictureBox1->Height / scale;
                        break;
                    }
                }

                if (fin==0&& arr[cageY + posCamY][cageX + posCamX] == ' ')
                {
```

```
                if (turn % 2 == 0)
                {
                        drawCross(clickX, clickY);
                        arr[cageY + posCamY][cageX + posCamX] = 'x';
                        label8->Text = "Ход: ноликов";
                        turn++;
                }
                else if (turn % 2 == 1)
                {
                        drawCircle(clickX, clickY);
                        arr[cageY + posCamY][cageX + posCamX] = 'o';
                        label8->Text = "Ход: крестиков";
                        turn++;
                }
        }

        fin = isFin();
        if (fin == 1)
        {
                label1->Text = "крестики победили";
        }
        else if (fin == 2)
        {
                label1->Text = "нолики победили";
        }
/*      for (int i = 0; i < sizeX; i++)
        {
                for (int j = 0; j < sizeY; j++)
                {
                        dataGridView1->Rows[i]->Cells[j]->Value = arr[i][j].ToString();
                }
        }*/

}
          void drawField()
          {
                  drawGrid();
                  for (int i = 0; i < sizeY; i++)
                  {
                          for (int j = 0; j < sizeX; j++)
                          {
                                  if (i >= posCamY && i <= posCamY + scale)
                                  {
                                          float cageX = (j - posCamX) * pictureBox1-
>Width / scale,
                                                  cageY = (i - posCamY) * pictureBox1-
>Height / scale;
                                          if (arr[i][j] == 'x')
                                          {
                                                  drawCross(cageX, cageY);
                                          }
                                          else if (arr[i][j] == 'o')
                                          {
```

```
                                        drawCircle(cageX, cageY);
                                    }
                                }

                            }
                        }
                }
        }
        void goRight() {
                if (posCamX < sizeX - scale)
                {
                        button3->Enabled = true;
                        posCamX++;
                        drawField();
                        return;
                }
                else if(posCamX >= sizeX - scale)
                {
                        button2->Enabled = false;
                }
        }
        void goLeft() {
                if (posCamX > 0)
                {

                        button2->Enabled = true;
                        posCamX--;
                        drawField();
                }
                else
                {
                        button3->Enabled = false;
                }
        }
        void goUp() {
                if (posCamY > 0)
                {
                        button5->Enabled = true;
                        posCamY--;
                        drawField();
                }
                else if(posCamY==0)
                {
                        button4->Enabled = false;
                }
        }
        void goDown()
        {
                if (posCamY < sizeY - scale)
                {
                        button4->Enabled = true;
                        posCamY++;
                        drawField();
                }
```

```
			else
			{
				button5->Enabled = false;
			}
		}
		void drawGrid() {
			g = pictureBox1->CreateGraphics();
			g->Clear(Color::White);
			int step = 0;
			Pen^ blackPen = gcnew Pen(Color::Black, 1);
			for (int step = 0; step < pictureBox1->Width; step += pictureBox1-
>Width / scale)
			{
				g->DrawLine(blackPen, step, 0, step, pictureBox1->Height);
			}
			step = 0;
			for (int step = 0; step < pictureBox1->Height; step += pictureBox1-
>Height / scale)
			{
				g->DrawLine(blackPen, 0, step, pictureBox1->Width, step);
			}

		}
		void showPosCam() {
			label2->Text = posCamX.ToString();
			label3->Text = posCamY.ToString();
		};
		void setCross(int i, int j)
		{
			arr[i][j] = 'x';
			float cageY=(i-posCamY)*pictureBox1->Height/scale;
			float cageX= (j - posCamX) * pictureBox1->Width / scale;
			drawCross(cageX, cageY);
		}
		void setCircle(int i, int j)
		{
			arr[i][j] = 'x';
			float cageY = (i - posCamY) * pictureBox1->Height / scale;
			float cageX = (j - posCamX) * pictureBox1->Width / scale;
			drawCircle(cageX, cageY);
		}
	private: System::Void Form1_Click(System::Object^ sender, System::EventArgs^ e) {
	}
	private: System::Void pictureBox1_LoadCompleted(System::Object^ sender,
System::ComponentModel::AsyncCompletedEventArgs^ e) {

	}
	private: System::Void pictureBox1_SizeChanged(System::Object^ sender,
System::EventArgs^ e) {
	}
	private: System::Void button1_TextChanged(System::Object^ sender,
System::EventArgs^ e) {
	}
```

```
	private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {

		label1->Text = "Кто же победит?";
		pictureBox1->Enabled = true;
		clearField();
	};
	private: System::Void button2_Click(System::Object^ sender, System::EventArgs^ e) {
		goRight();
		showPosCam();
	};
	private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e) {
		goLeft();
		showPosCam();
	};
	private: System::Void button4_Click(System::Object^ sender, System::EventArgs^ e) {
		goUp();
		showPosCam();
	}
	private: System::Void button5_Click(System::Object^ sender, System::EventArgs^ e) {
		goDown();
		showPosCam();
	}


	void clearField()
	{
		turn = 0;
		drawGrid();
		createArr(sizeX, sizeY);
	}
	private: System::Void radioButton2_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {

	}
private: System::Void radioButton1_CheckedChanged(System::Object^ sender,
System::EventArgs^ e) {
	clearField();
}
private: System::Void button6_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void textBox1_TextChanged(System::Object^ sender, System::EventArgs^ e) {
	if (textBox1->Text!="")
	{
		sizeX = Convert::ToInt32(textBox1->Text);
		sizeY = sizeX;
	}
	label7->Text = "x "+textBox1->Text->ToString();
}
private: System::Void label6_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void label2_Click(System::Object^ sender, System::EventArgs^ e) {
}
```

```
private: System::Void label1_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void label5_Click(System::Object^ sender, System::EventArgs^ e) {
}
private: System::Void trackBar1_Scroll(System::Object^ sender, System::EventArgs^ e) {
        label9->Text = trackBar1->Value.ToString();
        scale = trackBar1->Value;
}
};

}
```