### МИНОБРНАУКИ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

### «БЕЛГОРОДСКИЙ ГОСУДАРСТВЕНЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.Шухова»

**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

КУРСОВАЯ РАБОТА

По дисциплине: Сети ЭВМ и телекоммуникации

Тема: “ Разработка клиент-сервер приложения “Крестики-нолики” для сетей TCP/IP с использованием WinSock.”

Проверил: Федотов Е.А.

# Содержание

[Содержание 2](#_bookmark0)

[Введение 3](#_bookmark1)

[Глава 1. Протокол TCP, модель TCP/IP, WinSock 4](#_bookmark2)

* 1. [Краткая характеристика протокола TCP 4](#_bookmark3)
  2. [Краткая характеристика модели TCP/IP 5](#_bookmark4)
  3. [WinSock 6](#_bookmark5)

[Глава 2. Разработка программы 8](#_bookmark6)

* 1. [Описание модулей 8](#_bookmark7)
     1. [NetworkAPI. 8](#_bookmark8)
     2. [PInvoke 9](#_bookmark9)
  2. [Организация сетевого взаимодействия 10](#_bookmark10)
     1. [Формат передачи данных. 10](#_bookmark11)
     2. [Логика взаимодействия клиента и сервера. 11](#_bookmark12)
     3. [Устранение блокировок приложения связанных с winsock 12](#_bookmark13)
  3. [Структура проекта 15](#_bookmark14)

[Глава 3. Руководство пользователя 16](#_bookmark15)

* 1. [Введение 16](#_bookmark16)
  2. [Условия, при соблюдении которых обеспечивается применение ПО 16](#_bookmark17)
  3. [Подготовка к работе 16](#_bookmark18)
  4. [Порядок работы 17](#_bookmark19)
  5. [Аварийные ситуации. 17](#_bookmark20)

[Глава 4. Тестирование ПО 19](#_bookmark21)

[Скриншоты 20](#_bookmark22)

[Заключение 22](#_bookmark23)

[Список литературы 23](#_bookmark24)

[ПРИЛОЖЕНИЯ 24](#_bookmark25)

[Приложение А – Код сетевой игры. 24](#_bookmark26)

[Листинг A.1. – Листинг MainWindow.xaml.cs 24](#_bookmark27)

[Листинг A.2. – Листинг файла разметки MainWindow.xaml 29](#_bookmark28)

[Листинг A.3. – Листинг модуля NetworkAPI 34](#_bookmark29)

[Листинг A.4. – Листинг модуля PInvoke 36](#_bookmark30)

# Введение

Сегодня все крупные и успешные коммерческие приложения активно взаимодействуют с интернетом, эти приложения могут быть как веб, так и нативными, могут работать на различных протоколах (например, WebSocket) но в любом случае они основываются и работают на одном из основных протоколов передачи данных интернета, которым является TCP.

Во все время актуальность протокола TCP заключается в надёжной передаче данных между двумя удалёнными устройствами или программами, запущенным на одной локальной машине.

TCP соединения находят своё применение в абсолютно различных приложениях, например в современных системах рендеринга (например, V- Ray Distributed Rendering), для того, чтобы распараллеливать сложные вычисления между множеством мощных вычислительных машин и значительно ускорить процесс получения результата, для дальнейшей его обработки и не беспокоиться о надёжности передачи, так как TCP об этом позаботится.

В курсовой работе мы подробно рассмотрим протокол TCP, разработаем программу “Крестики-нолики” на языке программирования C#, для того, чтобы на примере показать сетевое TCP взаимодействие клиент-сервера и затем сделаем вывод о полученных результатах.

# Глава 1. Протокол TCP, модель TCP/IP, WinSock.

# Краткая характеристика протокола TCP

TCP (transmission control protocol) – «протокол управления передачей», предназначенный для управления передачей данных.

Разработка практически любого сетевого программного продукта как правило требует наличия сетевого протокола передачи данных, который позволит передать требуемые данные между программами.

Определить протокол для создания сетевого взаимодействия между программами следует осмысленно, так как есть TCP и UDP, и далеко не во всех случаях следует выбирать один и тот же протокол, например, для приложений, работающих в режиме “Реального времени” подошёл бы скорее UDP протокол, а для работы сервера лицензирования какого-нибудь коммерческого программного обеспечения идеально бы подошёл TCP протокол, так как там больше важна не скорость, а надёжность передачи.

Механизм TCP осуществляет установление надежного соединения между двумя устройствами и надежную передачу данных (потока байтов) от одной программы на некотором компьютере к другой программе на другом компьютере. Протокол TCP позволяет пересылать большие объемы информации, разделяя ее на отдельные части – пакеты, – которые доставляются получателю независимо. Также TCP отвечает за правильное восстановление данных из пакетов, полученных после передачи. В частности, TCP контролирует размер и целостность передаваемого пакета данных, автоматически осуществляя повторную пересылку при сбое передачи (при потере или повреждении данных), а также устраняет дублирование при получении двух копий одного пакета информации. Таким образом, TCP гарантирует целостность передаваемых данных и уведомление отправителя о результатах передачи.

Абсолютное ограничение на размер пакета TCP составляет 64 КБ (65535 байт), включая заголовки, то есть полезная передаваемая нагрузка порядком меньше, чем 65535 байт.

TCP пакет имеет следующую структуру:



Рис 1.1. Структура TCP пакета.

# Краткая характеристика модели TCP/IP

TCP/IP — сетевая модель передачи данных, представленных в цифровом виде. Модель описывает способ передачи данных от источника информации к получателю. В модели предполагается прохождение информации через четыре уровня, каждый из которых описывается правилом (протоколом передачи). Наборы правил, решающих задачу по передаче данных, составляют стек протоколов передачи данных, на которых базируется Интернет. Название TCP/IP происходит из двух важнейших протоколов семейства — Transmission Control Protocol (TCP) и Internet Protocol (IP), которые были первыми разработаны и описаны в данном стандарте.

Протоколы TCP/IP делятся на 4 уровня.

* Самый нижний (уровень IV) соответствует физическому и канальному уровням модели OSI. Этот уровень в протоколах TCP/IP не регламентируется, но поддерживает все популярные стандарты физического и канального уровня: для локальных сетей это Ethernet, Token Ring, FDDI, Fast Ethernet, 100VG-AnyLAN, для глобальных сетей - протоколы соединений «точка- точка» SLIP и PPP, протоколы территориальных сетей с коммутацией пакетов X.25, frame relay.
* Следующий уровень (уровень III) - это уровень межсетевого взаимодействия, который занимается передачей пакетов с использованием различных транспортных технологий локальных сетей, территориальных сетей, линий специальной связи и т.п.

В качестве основного протокола сетевого уровня (в терминах модели OSI) в стеке используется протокол IP, который изначально проектировался как протокол передачи пакетов в составных сетях, состоящих из большого количества локальных сетей, объединенных как локальными, так и глобальными связями. Поэтому протокол IP хорошо работает в сетях со

сложной топологией, рационально используя наличие в них подсистем и экономно расходуя пропускную способность низкоскоростных линий связи. Протокол IP является дейтаграммным протоколом, то есть он не гарантирует доставку пакетов до узла назначения, но старается это сделать.

К уровню межсетевого взаимодействия относятся и все протоколы, связанные с составлением и модификацией таблиц маршрутизации, такие как протоколы сбора маршрутной информации RIP (Routing Internet Protocol) и OSPF (Open Shortest Path First), а также протокол межсетевых управляющих сообщений ICMP (Internet Control Message Protocol). Последний протокол предназначен для обмена информацией об ошибках между маршрутизаторами сети и узлом

- источником пакета. С помощью специальных пакетов ICMP сообщается о невозможности доставки пакета, о превышении времени жизни или продолжительности сборки пакета из фрагментов, об аномальных величинах параметров, об изменении маршрута пересылки и типа обслуживания, о состоянии системы и т.п.

* Следующий уровень (уровень II) называется основным. На этом уровне функционируют протокол управления передачей TCP (Transmission Control Protocol) и протокол дейтаграмм пользователя UDP (User Datagram Protocol). Протокол TCP обеспечивает надежную передачу сообщений между удаленными прикладными процессами за счет образования виртуальных соединений. Протокол UDP обеспечивает передачу прикладных пакетов дейтаграммным способом, как и IP, и выполняет только функции связующего звена между сетевым протоколом и многочисленными прикладными процессами.
* Верхний уровень (уровень I) называется прикладным. За долгие годы использования в сетях различных стран и организаций стек TCP/IP накопил большое количество протоколов и сервисов прикладного уровня. К ним относятся такие широко используемые протоколы, как протокол копирования файлов FTP, протокол эмуляции терминала telnet, почтовый протокол SMTP, используемый в электронной почте сети Internet, гипертекстовые сервисы доступа к удаленной информации, такие как WWW и многие другие. Остановимся несколько подробнее на некоторых из них.

# WinSock

В вычислительной технике Windows Sockets API (WSA), позднее сокращенное до Winsock, представляет собой техническую спецификацию, определяющую, как сетевое программное обеспечение Windows должно получать доступ к сетевым службам, особенно к TCP/IP. Он определяет стандартный интерфейс между клиентским приложением Windows TCP/IP (таким как FTP-клиент или

веб-браузер) и лежащим в основе стека протоколов TCP/IP. Номенклатура основана на модели API сокетов Berkeley, используемой в BSD для обмена данными между программами.

Реализация Windows Sockets API располагается в библиотеке ws2\_32.dll и как правило располагается по следующему пути C:\Windows\System32\ws2\_32.dll. ws2\_32 предоставляет весь базовый набор функций для работы с сокетами: connect, recv, bind, send и т.д.

Весь список доступных функций можно просмотреть на MSDN (Microsoft Developer Network) либо в заголовочном файле Winsock.h

# Глава 2. Разработка программы

# Описание модулей.

Наша программа состоит из основного исходного файла mainwindow.xaml.cs и двух вспомогательных модулей, которые в основном связанны с сетевым взаимодействием между программами по TCP протоколу.

Рассмотрим все модули по подробнее.

# NetworkAPI.

### Описание модуля

NetworkAPI представляет собой объектно-ориентированный интерфейс для быстрой и удобной организации взаимодействия между клиентом и сервером.

### Спецификация модуля

Модуль NetworkAPI состоит из трёх классов:

### Packet

Назначение: представляет собой структуру пакета, которая будет использоваться при передаче.

Заголовок:

public class Packet

{

public string action; public object value;

}

### TCPServer

Назначение: этот объект представляет собой сервер, который позволяет при срабатывании конструктора запускать однопоточный асинхронный сервер, с возможностью поддержания соединения только с одним клиентом и установки колбэка на обработку подключения удалённого клиента и принятия входящих сообщений от клиента в произвольный момент времени. Отправка сообщений производится с помощью метода sendMessage, а остановка клиента с помощью метода stop.

Заголовок:

public class TCPServer

{

int s, s1;

public TCPServer(int port, Action<byte[]> RecieveCallback, Action<TCPServer> ConnectedCallback);

public bool sendMessage(string text); public void stop();

void waitConnection(IntPtr socket, Action<byte[]> RecieveCallback, Action<TCPServer> ConnectedCallback);

void waitMessages (IntPtr socket, Action<byte[]> RecieveCallback);

}

### TCPClient

Назначение: этот объект представляет собой клиент, который позволяет при срабатывании конструктора запускать асинхронный клиент и подключаться по IP и порту к удалённому серверу, с возможностью установки колбэка на обработку принятия входящих сообщений от сервера в произвольный момент времени. Отправка сообщений производится с помощью метода sendMessage, а остановка клиента с помощью метода stop.

Заголовок:

public class TCPClient

{

int s;

public TCPClient(string ip, int port, Action<byte[]> RecieveCallback); public bool sendMessage(string text);

public void stop();

void waitMessages(IntPtr socket, Action<byte[]> RecieveCallback);

}

# PInvoke.

### Описание модуля

PInvoke представляет собой модуль с интерфейсами для работы с ws2\_32.dll, так как некоторые вещи просто не могут быть выполнены в чистом .NET и наш модуль позволяет управляемому коду вызывать не управляемый код из вспомогательной системной библиотеки.

### Спецификация модуля

Pinvoke состоит исключительно только из прототипов функций и структур, описанных в библиотеке “ws2\_32.dll” и имеет следующий вид:

[DllImport("Ws2\_32.dll")]

public static extern int bind(IntPtr s, ref sockaddr\_in addr, int addrsize); [DllImport("Ws2\_32.dll")]

public static extern int listen(IntPtr s, int backlog); [DllImport("Ws2\_32.dll")]

public static extern IntPtr accept(IntPtr s, IntPtr addr, int addrsize);

…

# Организация сетевого взаимодействия.

Помимо того, что мы описали основные и необходимые функции для передачи данных по TCP протоколу, нам следует грамотно и однозначно сформулировать логику и правила передачи пакетов данных, одновременно учитывая тот фактор, что наша программа может быть запущена одновременно в нескольких экземплярах на одной и той же ЭВМ.

Следует так же отметить, что наша программа при запуске должна автоматически работать в режиме сервера, а при необходимости всегда иметь возможность перейти в режим клиента и подключиться по заданному адресу к серверу.

# Формат передачи данных.

Сперва следует определить структуру элементарного пакета, который позволил бы нам структурировать какие-то данные нашей предметной области таким образом, чтобы мы могли в будущем просто, удобно и однозначно передавать, и обрабатывать какие-то данные.

Наша предметная область связана с разработкой игры, а в игре всё основывается на состояниях и действиях. Состояния наших игроков будут изменяться под воздействием каких-либо действий (action), поэтому в нашем пакете следует выделить как минимум одно поле, которое будет удалённому клиенту сообщать какое действие стоит ему выполнить.

Ввиду того, что существует огромное множество ситуаций, когда только типа действия просто недостаточно для выполнения какого действия и требуется какая-то дополнительная информация, нам следует добавить ещё одно поле, которое будет являться полезной нагрузкой в нашем пакете и содержать информацию, которая будет необходима при выполнении действия определённого в пакете удалённым пользователем.

Передавать мы будем пакет данных следующей структуры:

public class Packet

{

public string action; public object value;

}

Где, Action – это действие, у нас будет три возможых действия: putX (означает, что в некоторую клетку ставится крест), putO (означает, что в некоторую клетку ставится ноль) и mylogin (используется для передачи логина соперника).

Value – это полезные передаваемые данные, тип зависит от контекста действия. Например, при передаче факта о постановке O или X (putO/putX) в value будет хранится объект с типом Position в котором будут хранится координаты x, y, а в случае передачи логина, в value будет храниться просто строка с логином.

Так как при передаче следует передавать исключительно массив байт, мы предварительно будем сериализовать наш объект Packet с помощью дополнительной библиотеки Newtonsoft.Json в строку, затем строку переводить в последовательность байт, а только потом передавать удалённому пользователю.

# Логика взаимодействия клиента и сервера.

Логику работы клиента и сервера описать достаточно просто. Наша программа запустившись автоматически запускает TCP сервер на первом свободном порту начиная с 5150 включительно, так как экземпляров программы на одном компьютере может быть запущенно несколько и вообще порт 5150 может быть занят какой-нибудь другой программой.

Так же учтём возможность подключения нашей программы к удалённому серверу, но стоит отметить, что сперва следует остановить TCP сервер для предотвращения проблем в дальнейшем.

При соединении клиента и сервера, клиент и сервер должны будут отправить друг другу свои никнеймы заданные изначально при запуске программы, для того, чтобы можно было во время игры вести игровой счёт и видеть, какому игроку какой счёт соответствует.

Далее происходит сама игра. Определим правило, что первый игрок, который будет ходить и будет в протяжении всей игровой сессии крестиком это будет сервер. В протяжении всей игры поочерёдно клиент будет отправлять серверу, а сервер клиенту пакет с действием о постановке крестика или нолика (putX, putO), который программа должна моментально же выполнить.

Всю логику можно показать на следующей схеме:

...



Сервер Клиент

TCP подключение

mylogin

mylogin

putX

putO

putX

putO

putX

putO

Рис 2.1. Схема алгоритма взаимодействия между сервером и клиентом.

...

# Устранение блокировок приложения связанных с winsock.

Так как мы работаем с WinSock API, мы должны знать, что некоторая часть функций из библиотеки winsock являются блокирующимся, в частности:

* + - * accept() - создает новый сокет и подключает его к удаленному компьютеру
      * closesocket() - закрывает одну из сторон соединения
      * connect() - инициализирует соединение со стороны указаннного сокета
      * recv() - принимает данные от подключенного сокета
      * recfrom() - принимает данные от подлюченного или неподключенного сокета
      * select() - управеление состоянием нескольких сокетов
      * send() - посылает данные подключенному сокету
      * sendto() - посылает данные подлюченному или неподключенному сокету

Блокировка функции приводит к такой ситуации, что, например, при приёме данных, наш основной поток выполнения программы будет постоянно ожидать тот момент, когда наша функция примет какие-либо данные, а так как наша программа будет активно читать и отправлять данные, то, следовательно,

большую часть времени наша программа будет в зависнувшем состоянии и выдавать сообщение, что наша программа не отвечает.

Эту проблему можно решить просто, выполняя функции, связанные с отправлением и прослушкой входящих сообщений в асинхронном режиме.

На примере сервера можно рассмотреть, как мы будем устранять блокировки программы за счёт асинхронности:

public TCPServer(int port, Action<byte[]> RecieveCallback, Action<TCPServer> ConnectedCallback)

{

…

if (result == SUCCESS)

{

…

/\* Делаем асинхронный вызов функции, постоянно ожидающей подключения удалённого клиента. \*/

new Task(() => { waitConnection(new IntPtr(s), RecieveCallback, ConnectedCallback); }).Start();

}

}

…

void waitConnection(IntPtr socket, Action<byte[]> RecieveCallback, Action<TCPServer> ConnectedCallback){

s1 = (int)accept(new IntPtr(s), new IntPtr(0), 0); // Блокируется

/\* Делаем вызов Callback функции, для того чтобы наша программа была в курсе о том,

что был подключен новый клиент. Асинхронность реализована для того, что в случае если у нас в callback какая-нибудь блокирующая функция, например, открытие диалогового окна для сохранения файла,

то наша сетевая часть программы не повисла и следовательно вся наша программа.\*/

new Task(() => ConnectedCallback(this)).Start();

/\* Делаем асинхронный вызов функции, работающей в бесконечном цикле и постоянно принимающей новые сообщения от подключившегося клиента. \*/

new Task(() => waitMessages(new IntPtr(s1), RecieveCallback)).Start();

}

void waitMessages(IntPtr socket, Action<byte[]> RecieveCallback){

byte[] buffer = new byte[256];

while (true)

if (recv(socket, buffer, 256, 0) > 0) // Блокируется, но уже не в основном

потоке

/\* Вызов callback функции отвечающей за обработку пришедших данных. \*/

RecieveCallback(buffer);

}

accept

recv

-socket:int

-waitConnection( IntPtr, Action<byte[]>, Action<TCPServer>)

-waitMessages( IntPtr, Action<byte[]>)

+TCPServer(int,Action<byte[]>,Action<T CPServer> )

+sendMessage(string)

+stop()

TCPServer

RecieveCallback

ConnectedCallback

Рис 2.2. Упрощённая схема описания нашего TCP сервера

Таким образом мы реализовали абсолютно асинхронный не блокирующийся TCP сервер, который будет при необходимости сообщать нашей основной программе посредством Callback функций о подключенных новых пользователях и пришедших от них сообщений.

# Структура проекта

Наш проект будет состоять из следующих файлов:

|  |  |
| --- | --- |
| Название | Назначение |
| Kursach.sln | Файлы проекта с информацией о том:   * Как построить проект. * Настройки проекта. * О сборках, на которые ссылается проект. |
| Kursach.csproj |
| MainWindow.xaml | Разметка нашего пользовательского интерфейса. |
| MainWindow.xaml.cs | Исходный код нашей логики программы. |
| Pinvoke.cs | Вспомогательные модули. |
| NetworkAPI.cs |

И следующих файлов ресурсов:

|  |  |
| --- | --- |
| Название | Назначение |
| tic-tac-toe.ico | Ico значок, иконка программы |
| win.wav | Wav аудиозвук, который будет воспроизводиться при нашей победе. |
| lose.wav | Wav Аудиозвук, который будет воспроизводиться при нашем проигрыше. |
| click.wav | Wav Аудиозвук, который будет воспроизводиться при открытии крестика или нолика. |
| icon-crown.png | Png изображение, которое будет обозначать лидирующего игрока. |

Также из вспомогательных сторонних дополнительных библиотек будем использовать Json.NET (Newtonsoft), который позволяет сериализовать/ десириализовать данные в Json формат и установим его с помощью, встроенной в VisualStudio системы управления пакетами NuGet.

* 1. **Введение**

# Глава 3. Руководство пользователя.

Область применения данного ПО ориентирована на проведение досуга, а также образовательную деятельность, связанную с компьютерной грамотностью.

Программа “Крестики-нолики” предназначена для организации игрового процесса “крестики-нолики” между партнёрами по сети, как локальной, так и глобальной.

Пользователю для игры, достаточно знать и иметь базовые навыки работы с компьютером, ни каких специальных знаний не требуется.

* 1. **Условия, при соблюдении которых обеспечивается применение ПО.**

Набор условий, необходимых для возможности запуска и работы программного продукта, следующий:

|  |  |
| --- | --- |
| Операционная система | Windows 7/Vista/8.0/8.1/10 |
| Процессор | 1.6 Ghz |
| Оперативная память | 2 Гб DDR3 RAM |
| Видеоадаптер | Intel hd graphics |
| Свободное место на HDD | 30 Мб |
| Оптический накопитель | CD-ROM (требуется для установки) |
| Устройства взаимодействия с пользователем | Мышь, клавиатура |
| Другие устройства | Звуковая карта, колонки и/или наушники |
| Наличие следующих программных пакетов | .NET Framework 4 |

### Подготовка к работе

Для работы с программой, необходимо сперва иметь архив с исполняемым файлом нашего программного обеспечения, заранее скачав из интернет ресурса “Яндекс диск” или “Google диск”, или при отсутствии интернета просто скопировав экземпляр программы с переносимого флэш носителя.

После загрузки архива с нашим ПО, необходимо zip архив распаковать с помощью встроенных в систему средств распаковки zip архивов на диск, в котором будет доступно свободно как минимум 10 МБ памяти.

Перед началом запуска нашей программы следует проверить интернет соединение, которое необходимо для взаимодействия игроков по сети интернет.

Выполнив все подготовительные пункты, можно запустить исполняемый файл нашей программы.

### Порядок работы.

Порядок работы с программой следующий:

* + 1. Авторизация – запустив программу, пользователю необходимо указать в строке, под надписью: “Введите ваш ник:” свой ник (игровое прозвище), для того чтобы напарник мог понять, с кем он играет.

В случае если пользователь будет желать подключения напарника к нему, то пользователь должен запомнить свой игровой порт, который закреплён за текущим запущенным экземпляром игры и определяется программой автоматически при каждом запуске программы.

После того как пользователь введёт свой ник, он должен нажать на кнопку “играть” и в результате перейти на неактивное игровое поле.

* + 1. Подключение – для самой игры пользователь должен подключиться к своему напарнику введя в строку “Адрес”, ip адрес удалённого компьютера и игровой порт (по стандарту он равен 5150), который заранее пользователь должен узнать у своего напарника.
    2. Игра – после установки подключения, в игре будет активировано игровое поле, по которому можно нажимать и устанавливать крестики или нолики как по правилам “крестиков-ноликов”, за исключением того, что для победы требуется не 3 фигуры в линию, а 5.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| После | победы | одного | из | игроков, будет | воспроизведён | |
| сигнализирующий о победе игрока звук или напротив о проигрыше. Для | | | | | | |
| просмотра счёта, необходимо посмотреть в правую верхнюю часть окна | | | | | | |
| и увидеть количество побед, а также увидеть кто лидирует. | | | | | |  |

* + 1. Завершение игры – для завершения игры достаточно нажать на крестик в самом правом верхнем углу окна программы и закрыть запущенный экземпляр программы.

### Аварийные ситуации.

* Действия в случае продолжительного зависания программы: Шаг 1. Запустить системный диспетчер задач.

Шаг 2. Закрыть процесс “Крестики-нолики”.

* Действия в случае долгого отсутствия ответа от напарника:

Шаг 1. Закрыть ПО.

Шаг 2. Отключить интернет соединение (в случае, если оно включено).

Шаг 3. Подождать 2-3 минуты.

Шаг 4. Включить интернет.

Шаг 5. Запустить ПО.

Шаг 6. Продолжать работу с ПО.

* Действия в случае, когда при запуске программы абсолютно ничего не появляется и не происходит:

Шаг 1. Установить Microsoft .NET Framework 4.8 либо более поздней версии.

Шаг 2. Перезапустить компьютер.

# Глава 4. Тестирование ПО.

Тестирование программного обеспечения (ПО) - это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. В более широком смысле, тестирование ПО - это техника контроля качества программного продукта и анализ полученных результатов в результате тестирования.

Для проверки нашей уже скомпилированной программы на наличие ошибки, мы проведём 10 тестовых и 5 контрольных запусков нашей программы в соответствии с руководством пользователя, описанным в 3-ей главе и в конце, на основе полученных результатов сделаем вывод.

В случае возникновения аварийной ситуации на контрольном тесте, произведём правки исходного кода нашей программы и проведём тестовый цикл заново.

|  |  |  |
| --- | --- | --- |
| Номер запуска | Ошибки | Статус |
| 1 | Отсутствуют | Тест |
| 2 | Отсутствуют | Тест |
| 3 | Отсутствуют | Тест |
| 4 | Отсутствуют | Тест |
| 5 | Отсутствуют | Тест |
| 6 | Отсутствуют | Тест |
| 7 | Отсутствуют | Тест |
| 8 | Отсутствуют | Тест |
| 9 | Отсутствуют | Тест |
| 10 | Отсутствуют | Тест |
| 11 | Отсутствуют | Контроль |
| 12 | Отсутствуют | Контроль |
| 13 | Отсутствуют | Контроль |
| 14 | Отсутствуют | Контроль |
| 15 | Отсутствуют | Контроль |

Рис 4.1. Результаты проведённых тестов.

В результате проведённых тестов ошибок выявлено не было, все алгоритмы были отработаны правильно, звуковые эффекты и графические элементы были проверены, а также были проверены возможности сетевой игры как в локальной, так и глобальной сети интернет.

В заключении можно сделать вывод, что в результате успешного тестирования нашего ПО, можно переходить к процедуре внедрения программного обеспечения в эксплуатацию.

# Скриншоты

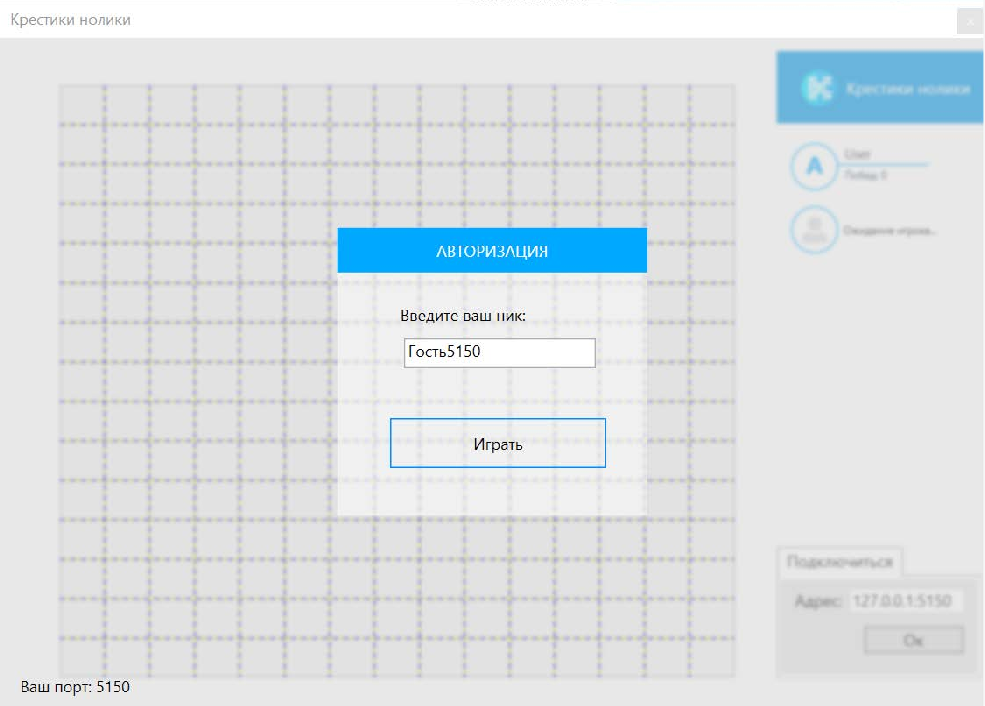


Рис 5.1. Скриншот программы. Форма авторизация.

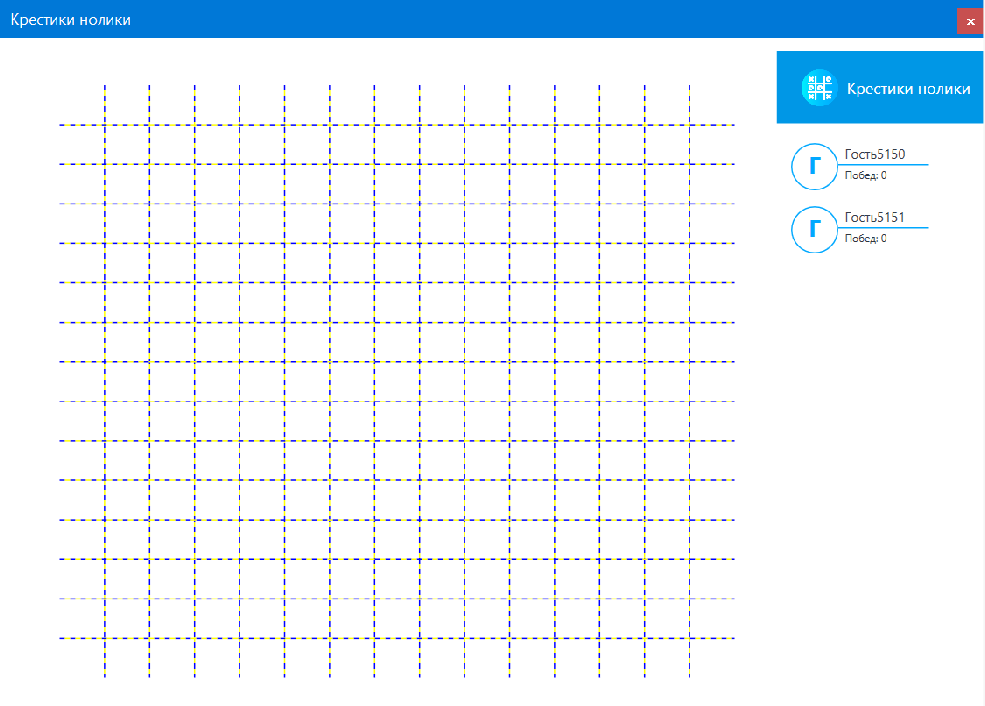


Рис 5.2. Скриншот программы. Игровое поле.

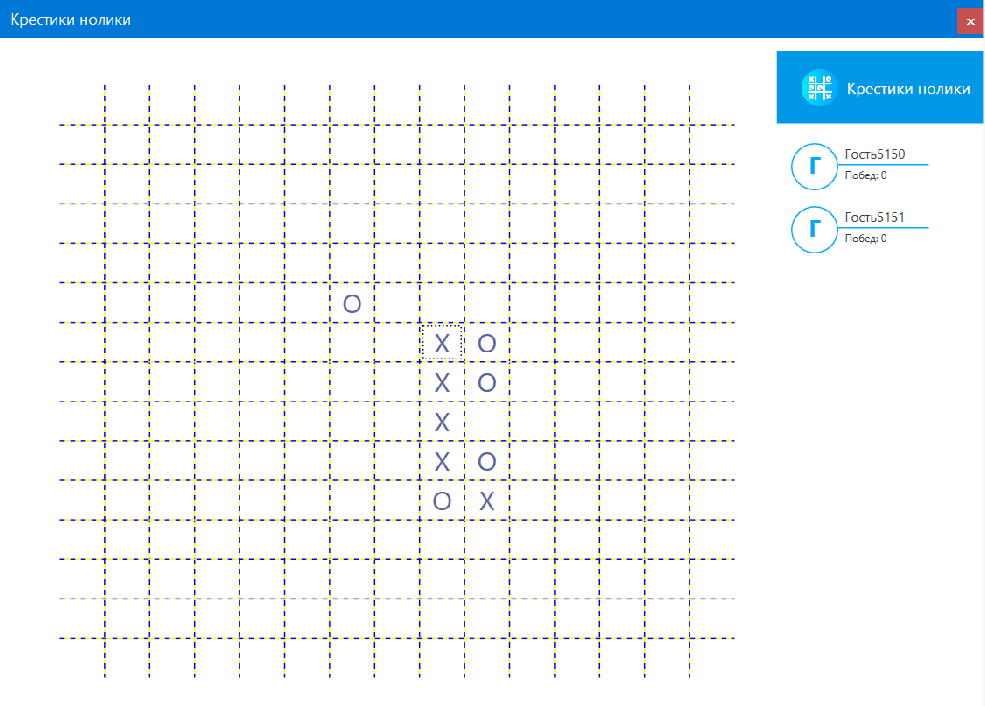


Рис 5.3. Скриншот программы. Игровой процесс.

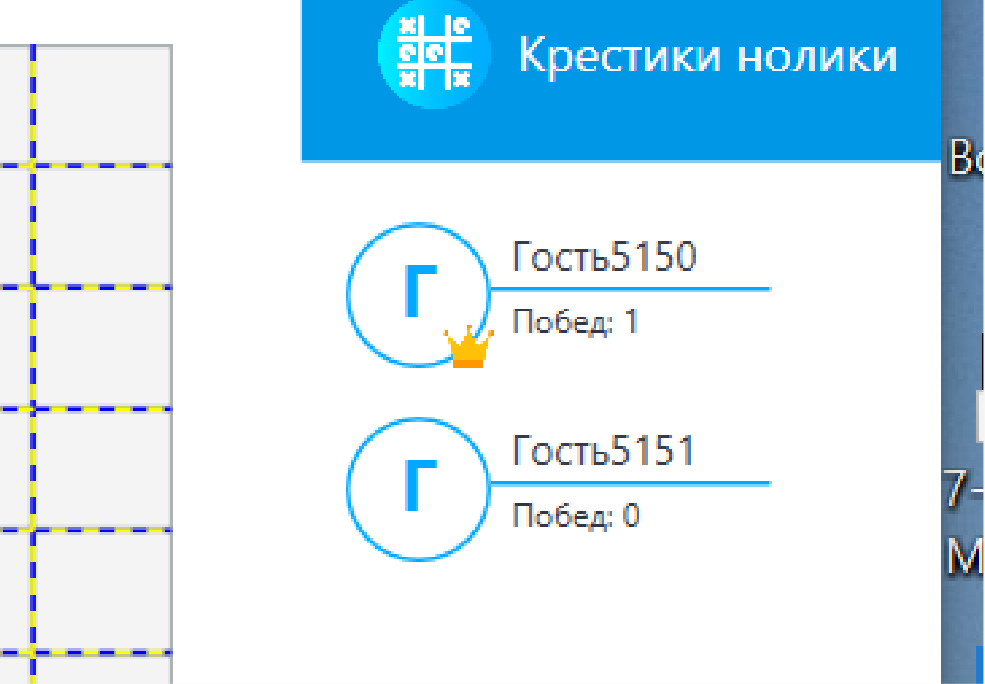


Рис 5.4. Скриншот программы. Индикатор лидирующего игрока.

# Заключение

В данной работе мы подробно разобрали, что такое Winsock и TCP/IP, где этот протокол сегодня используют и вообще следует использовать, а также его особенности, после краткого теоретического материала мы описали структуру нашей будущей программы в виде спецификаций, по которым далее мы разработали программу “крестики-нолики”, написав её на языке C# с графической подсистемой Windows Presentation Foundation (WPF) для построения визуально привлекательного пользовательского интерфейса.

Следует отметить, что наша программа, имеет возможность работать имеет возможность в сети, как локальной, так и глобальной. Сетевое взаимодействие между приложениями происходит с помощью протокола передачи данных TCP и для реализации собственно самого взаимодействия, мы использовали API системной библиотеки ws2\_32.lib (Windows Sockets API), который представляет собою интерфейс между нашим приложением и транспортным протоколом.

Написав и скомпилировав нашу программу, мы провели её тестирование, устранили недостатки и получили конечный продукт, который является результатом нашей курсовой работы и показывает, что мы освоили материал, связанный с протоколами и позволяющий нам разрабатывать сетевые приложения.

# Список литературы

1. Олифер В.Г., Олифер Н.А. Компьютерные сети. Принципы, технологии, протоколы (учебник для ВУЗ’ов). —СПб.: Питер, 2011. —944 c.
2. Э.Таненбаум, Д.Уэзеролл. Компьютерные сети. —СПб.: Питер, 2012. —960 c.
3. Исаченко О.В.. Программное обеспечение компьютерных сетей. —M.:

Инфра-М, 2012. —120 c.

1. Смелянский Р.Л. Компьютерные сети (в 2 томах). —M.: Академия, 2011 (том 1: Системы передачи данных, —304 c.; том 2: Сети ЭВМ, —240 с.).
2. Баканов В.М. Сетевые технологии: учебное пособие (курс лекций). —M.:

МГУПИ, 2008. –105 c.

1. Терри Оглтри. Модернизация и ремонт сетей = Upgrading and Repairing Networks. — 4-е изд. — М.: «Вильямс», 2005. — С. 1328. — ISBN 0-7897-2817- 6.
2. Дуглас Камер. Сети TCP/IP, том 1. Принципы, протоколы и структура = Internetworking with TCP/IP, Vol. 1: Principles, Protocols and Architecture. — М.:

«Вильямс», 2003. — С. 880. — ISBN 0-13-018380-6.

1. Семенов Ю. А. Протоколы Internet. — 2-е изд., стереотип.. — М.: Горячая линия - Телеком, 2005. — 1100 с. — 1150 экз. — ISBN 5-93517-044-2.
2. Паркер Т., Сиян К. TCP/IP. Для профессионалов. — 3-е изд.. — СПб.: Питер,

2004. — 859 с. — 4000 экз. — ISBN 5-8046-0041-9.

# ПРИЛОЖЕНИЯ

# Приложение А – Код сетевой игры.

## Листинг A.1. – Листинг MainWindow.xaml.cs

using Newtonsoft.Json; using Newtonsoft.Json.Linq; using System;

using System.Collections;

using System.Collections.Generic; using System.Linq;

using System.Media; using System.Text;

using System.Threading.Tasks; using System.Windows;

using System.Windows.Controls; using System.Windows.Data;

using System.Windows.Documents; using System.Windows.Input;

using System.Windows.Markup; using System.Windows.Media;

using System.Windows.Media.Imaging; using System.Windows.Navigation; using System.Windows.Shapes;

using static Kursach.NetworkAPI; #pragma comment(lib, "Ws2\_32.lib")

namespace Kursach

{

public partial class MainWindow : Window

{

TCPServer tcpserver;

TCPClient tcpclient;

string player = "X";

int win1 = 0, win2 = 0;

public MainWindow() { InitializeComponent();

}

class Position

{

public int \_x, \_y;

public Position(int x, int y) { \_x = x; \_y = y; }

public int getX() { return \_x; }

public int getY() { return \_y; }

public void setX(int x) { \_x = x; }

}

}

{ \_y = y;

public void setY(int y)

public void playwav(string filename)

{

var resource = Application.GetResourceStream(new Uri(filename, UriKind.Relative));

if (resource != null)

{

SoundPlayer player = new SoundPlayer(resource.Stream);

player.Load();

player.Play();

}

}

// Событие срабатываемое при нажатии на одну из ячеек поля ->

private void buttonClickEvent(object sender, EventArgs e)

{

Console.WriteLine(string.Format("clicked {0}/{1}", (((Button)sender).Tag as Position).getX(), (((Button)sender).Tag as Position).getY()));

((Button)sender).Content = tcpserver != null ? "X" : "O";

if (tcpserver != null) tcpserver.sendMessage(JsonConvert.SerializeObject(new Packet() { action = "putX", value = ((Button)sender).Tag }));

if (tcpclient != null) tcpclient.sendMessage(JsonConvert.SerializeObject(new Packet() { action = "putO", value = ((Button)sender).Tag }));

Task.Factory.StartNew(() => playwav("click.wav"));

if (checkWin((((Button)sender).Tag as Position).getX(), (((Button)sender).Tag as Position).getY(), player))

{

Task.Factory.StartNew(() => playwav("win.wav"));

Dispatcher.Invoke(() =>

{

wins1.Content = "Побед: " + (++win1).ToString();

RestartGame();

});

}

GameTable.IsEnabled = false;

}

// Функция, позволяющая находить объект с заданным в аргументе childName именем

public T FindChild<T>(DependencyObject depObj, string childName) where T : DependencyObject

{

if (depObj == null) return null;

if (depObj is T && ((FrameworkElement)depObj).Name == childName) return depObj as T;

int childcounts = VisualTreeHelper.GetChildrenCount(depObj);

for (int i = 0; i < VisualTreeHelper.GetChildrenCount(depObj); i++)

{

DependencyObject child = VisualTreeHelper.GetChild(depObj, i);

T obj = FindChild<T>(child, childName);

if (obj != null) return obj;

}

return null;

}

bool checkWin(int x, int y, string gamer\_symbol)

{

int sH = 1, sV = 1, sD1 = 1, sD2 = 1;

Dispatcher.Invoke(() =>

{

for (int tmpx = x + 1; tmpx < 15 && FindChild<Button>(this.GameTable, "btn" + tmpx.ToString() + y.ToString()).Content.ToString().Contains(gamer\_symbol); tmpx++, sH++) ;

for (int tmpx = x - 1; tmpx > 0 && FindChild<Button>(this.GameTable, "btn" + tmpx.ToString() + y.ToString()).Content.ToString().Contains(gamer\_symbol); tmpx--, sH++) ;

for (int tmpy = y + 1; tmpy < 15 && FindChild<Button>(this.GameTable, "btn" + x.ToString() + tmpy.ToString()).Content.ToString().Contains(gamer\_symbol); tmpy++, sV++) ;

for (int tmpy = y - 1; tmpy > 0 && FindChild<Button>(this.GameTable, "btn" + x.ToString() + tmpy.ToString()).Content.ToString().Contains(gamer\_symbol); tmpy--, sV++) ;

for (int tmpx = x + 1, tmpy = y + 1; (tmpx < 15 && tmpy < 15) && FindChild<Button>(this.GameTable, "btn" + tmpx.ToString() + tmpy.ToString()).Content.ToString().Contains(gamer\_symbol); tmpy++, tmpx++, sD1++) ;

for (int tmpx = x - 1, tmpy = y - 1; (tmpx > 0 && tmpy > 0 ) && FindChild<Button>(this.GameTable, "btn" + tmpx.ToString() + tmpy.ToString()).Content.ToString().Contains(gamer\_symbol); tmpy--, tmpx--, sD1++) ;

for (int tmpx = x + 1, tmpy = y - 1; (tmpx < 15 && tmpy > 0 ) && FindChild<Button>(this.GameTable, "btn" + tmpx.ToString() + tmpy.ToString()).Content.ToString().Contains(gamer\_symbol); tmpx++, tmpy--, sD2++) ;

for (int tmpx = x - 1, tmpy = y + 1; (tmpx > 0 && tmpy < 15) && FindChild<Button>(this.GameTable, "btn" + tmpx.ToString() + tmpy.ToString()).Content.ToString().Contains(gamer\_symbol); tmpx--, tmpy++, sD2++) ;

});

if (sH >= 5 || sV >= 5 || sD1 >= 5 || sD2 >= 5) return true;

return false;

}

void RestartGame()

{

for (int i = 0; i < 15; i++) for (int j = 0; j < 15; j++) Dispatcher.Invoke(() => (FindChild<Button>(this.GameTable, "btn" + i.ToString() + j.ToString())).Content = "");

if (win1 > win2)

{

crown1.Visibility = Visibility.Visible;

crown2.Visibility = Visibility.Collapsed;

}

else if (win1 < win2)

{

crown1.Visibility = Visibility.Collapsed;

crown2.Visibility = Visibility.Visible;

}

else

{

crown1.Visibility = Visibility.Collapsed;

crown2.Visibility = Visibility.Collapsed;

}

}

void ReceivedBuffer(byte[] buffer)

{

var data = Encoding.UTF32.GetString(buffer).Replace("\0", "").Replace("}}}", "}}");

JObject jObject = JObject.Parse(data.Replace("}}}", "}}"));

if (jObject["action"].Value<String>().Contains("put"))

{

Console.WriteLine("btn" + jObject["value"]["\_x"].Value<int>().ToString() + jObject["value"]["\_y"].Value<int>().ToString());

Dispatcher.Invoke(() =>

{

(FindChild<Button>(this.GameTable, "btn" + jObject["value"]["\_x"].Value<int>().ToString() + jObject["value"]["\_y"].Value<int>().ToString())).Content = jObject["action"].Value<String>()

== "putO" ? "O" : "X";

GameTable.IsEnabled = true;

});

if (checkWin(jObject["value"]["\_x"].Value<int>(), jObject["value"]["\_y"].Value<int>(), jObject["action"].Value<String>() == "putO" ? "O" : "X"))

{

Task.Factory.StartNew(() => playwav("gg.wav"));

Dispatcher.Invoke(() =>

{

wins2.Content = "Побед: " + (++win2).ToString();

RestartGame();

});

}

}

else

{

Dispatcher.Invoke(() =>

{

lblname2.Content = jObject["value"].Value<string>();

shortlabel2.Content = jObject["value"].Value<string>()[0];

if (tcpserver != null) tcpserver.sendMessage(JsonConvert.SerializeObject(new Packet() { action = "mylogin", value = loginedit.Text }));

waitframe.Visibility = Visibility.Collapsed;

});

}

}

void Connected(TCPServer sender)

{

Dispatcher.Invoke(() =>

{

waitframe.Visibility = Visibility.Collapsed;

sender.sendMessage(JsonConvert.SerializeObject(new Packet() { action = "mylogin", value = loginedit.Text }));

GameTable.IsEnabled = true;

tabControl.Visibility = Visibility.Collapsed;

});

}

private void Window\_Initialized(object sender, EventArgs e)

{

for (int i = 0; i < 15; i++)

for (int j = 0; j < 15; j++)

{

var tmp = new Button() { Name = "btn" + j.ToString() + i.ToString(), Tag = new Position(j, i), Content = "", Background = Brushes.Transparent, BorderBrush = null, FontSize = 20, Padding = new Thickness(0, -1, 0, 0), Foreground = (Brush)(new BrushConverter()).ConvertFromString("#5d61a2") };

tmp.Click += new RoutedEventHandler(buttonClickEvent);

Grid.SetRow (tmp, i);

Grid.SetColumn(tmp, j);

GameTable.Children.Add(tmp);

}

int port = 5150;

while (tcpserver == null)

try

{

tcpserver = new TCPServer(port, ReceivedBuffer, Connected); portLbl.Content =

"Ваш порт: " + port.ToString();

}

catch (Exception)

{

port++;

}

player = "X";

loginedit.Text = "Гость" + port.ToString();

}

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

{

tcpserver.stop();

tcpserver = null;

tcpclient = new TCPClient(addrTextBox.Text.Split(':')[0], addrTextBox.Text.Split(':').Length == 1 ? 5150 : int.Parse(addrTextBox.Text.Split(':')[1]), ReceivedBuffer);

tcpclient.sendMessage(JsonConvert.SerializeObject(new Packet() { action = "mylogin", value = loginedit.Text }));

Dispatcher.Invoke(() =>

{

GameTable.IsEnabled = false;

tabControl.Visibility = Visibility.Collapsed;

});

player = "O";

}

private void button1\_Click(object sender, RoutedEventArgs e)

{

loginframe.Visibility = Visibility.Collapsed;

gameframe.Effect = null;

lblname1.Content = loginedit.Text;

shortlabel1.Content = loginedit.Text[0];

}

}

}

## Листинг A.2. – Листинг файла разметки MainWindow.xaml

<Window x:Class="Kursash.MainWindow"

xmlns=["http://s](http://schemas.microsoft.com/winfx/2006/xaml/presentation)c[hemas.microsoft.com/winfx/2006/xaml/presentation](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"](http://schemas.microsoft.com/expression/blend/2008)

xmlns:mc="[http://schemas.openxmlformats.org/markup-compatibility/2006"](http://schemas.openxmlformats.org/markup-compatibility/2006)

xmlns:xf="clr-namespace:XamlFlair;assembly=XamlFlair.WPF"

xmlns:local="clr-namespace:Kursash"

mc:Ignorable="d"

Title="MainWindow" Height="545.374" Width="764.201" Initialized="Window\_Initialized">

<Grid>

<Grid x:Name="gameframe">

<Grid.Effect>

<BlurEffect Radius="0" RenderingBias="Quality"/>

</Grid.Effect>

<Grid Name="GameTable" ShowGridLines="True" Margin="0,36,190.333,0"

HorizontalAlignment="Right" Width="514" Height="451" VerticalAlignment="Top" IsEnabled="False">

<Grid.RowDefinitions>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

</Grid>

<Grid HorizontalAlignment="Left" Height="36" Margin="602,80,0,0" VerticalAlignment="Top" Width="108">

<Ellipse HorizontalAlignment="Left" Height="36" VerticalAlignment="Top" Width="36" Stroke="#FF00A8FF"/>

<Label x:Name="shortlabel1" Content="А" HorizontalAlignment="Left" VerticalAlignment="Top" Background="{x:Null}" Foreground="#FF00A8FF" Width="36" Height="36" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Padding="0,0,0,4" FontWeight="Bold" FontSize="18"/>

<Label x:Name="lblname1" Content="User" HorizontalAlignment="Left"

Margin="41,1,0,0" VerticalAlignment="Top" FontSize="10" Height="13" Width="64"

Padding="0" Foreground="#FF353B48"/>

<Rectangle HorizontalAlignment="Left" Height="1" Margin="36,16,0,0" VerticalAlignment="Top" Width="69" Fill="#FF00A8FF" RenderTransformOrigin="0.5,0.5">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleY="-1"/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

<Label x:Name="wins1" Content="Побед: 0" HorizontalAlignment="Left" Margin="41,19,0,0" VerticalAlignment="Top" FontSize="8" Height="11" Padding="0" Foreground="#FF2F3640" Width="64"/>

<Image x:Name="crown1" Margin="23,23,70,-2" Source="icons8-crown-96.png" Stretch="Fill" Width="15" Height="15" Visibility="Collapsed"/>

</Grid>

<Grid HorizontalAlignment="Left" Height="36" Margin="602,128,0,0" VerticalAlignment="Top" Width="108" Visibility="Hidden">

<Ellipse HorizontalAlignment="Left" Height="36" VerticalAlignment="Top" Width="36" Stroke="#FF00A8FF" Visibility="Hidden"/>

<Label x:Name="label\_Copy1" Content="А" HorizontalAlignment="Left" VerticalAlignment="Top" Background="{x:Null}" Foreground="#FF00A8FF" Width="36" Height="36" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Padding="0,0,0,4" FontWeight="Bold" FontSize="18" Visibility="Hidden"/>

<Label x:Name="label2" Content="Александр" HorizontalAlignment="Left" Margin="41,1,0,0" VerticalAlignment="Top" FontSize="10" Height="13" Width="64" Padding="0" Foreground="#FF353B48" Visibility="Hidden"/>

<Rectangle HorizontalAlignment="Left" Height="1" Margin="36,16,0,0" VerticalAlignment="Top" Width="69" Fill="#FF00A8FF" RenderTransformOrigin="0.5,0.5" Visibility="Hidden">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleY="-1"/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

<Label x:Name="label3" Content="Побед:" HorizontalAlignment="Left" Margin="41,19,0,0" VerticalAlignment="Top" FontSize="8" Height="11" Padding="0" Foreground="#FF2F3640" Width="64" Visibility="Hidden"/>

</Grid>

<Rectangle Fill="#FF0097E6" HorizontalAlignment="Left" Height="55" Margin="591,10,-1.667,0" VerticalAlignment="Top" Width="168"/>

<Label x:Name="label4" Content="Крестики нолики" HorizontalAlignment="Left" Margin="591,10,-1.667,0" VerticalAlignment="Top" Foreground="White" Width="168" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Height="55"/>

<Image x:Name="icons8\_crown\_96\_png\_Copy" Margin="624,149,118.333,352" Source="icons8-crown-96.png" Stretch="Fill" Visibility="Hidden"/>

<TabControl x:Name="tabControl" HorizontalAlignment="Left" Height="100" Margin="591,387,0,0" VerticalAlignment="Top" Width="156">

<TabItem Header="Подключиться">

<Grid Background="#FFE5E5E5">

<TextBox x:Name="addrTextBox" HorizontalAlignment="Left" Height="17" Margin="33,8,0,0" TextWrapping="Wrap" Text="192.168.1.1" VerticalAlignment="Top" Width="106" BorderThickness="0"/>

<Label x:Name="label6" Content="IP:" HorizontalAlignment="Left" Margin="10,4,0,0" VerticalAlignment="Top" Padding="0" Height="23" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Width="23"/>

<Button x:Name="button" Content="Ок" HorizontalAlignment="Left"

Margin="64,36,0,0" VerticalAlignment="Top" Width="75" Click="button\_Click"/>

</Grid>

</TabItem>

</TabControl>

<Grid HorizontalAlignment="Left" Height="36" Margin="602,128,0,0" VerticalAlignment="Top" Width="108">

<Ellipse HorizontalAlignment="Left" Height="36" VerticalAlignment="Top" Width="36" Stroke="#FF00A8FF"/>

<Label x:Name="shortlabel2" Content="А" HorizontalAlignment="Left" VerticalAlignment="Top" Background="{x:Null}" Foreground="#FF00A8FF" Width="36" Height="36" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Padding="0,0,0,4" FontWeight="Bold" FontSize="18"/>

<Label x:Name="lblname2" Content="User" HorizontalAlignment="Left" Margin="41,1,0,0" VerticalAlignment="Top" FontSize="10" Height="13" Width="64" Padding="0" Foreground="#FF353B48"/>

<Rectangle HorizontalAlignment="Left" Height="1" Margin="36,16,0,0" VerticalAlignment="Top" Width="69" Fill="#FF00A8FF" RenderTransformOrigin="0.5,0.5">

<Rectangle.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleY="-1"/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Rectangle.RenderTransform>

</Rectangle>

<Label x:Name="wins2" Content="Побед: 0" HorizontalAlignment="Left" Margin="41,19,0,0" VerticalAlignment="Top" FontSize="8" Height="11" Padding="0" Foreground="#FF2F3640" Width="64"/>

<Image x:Name="crown2" Margin="23,23,70,-2" Source="icons8-crown-96.png" Stretch="Fill" Width="15" Height="15" Visibility="Collapsed"/>

</Grid>

<Grid x:Name="waitframe" HorizontalAlignment="Left" Height="36" Margin="602,128,0,0" VerticalAlignment="Top" Width="115" Background="White">

<Ellipse HorizontalAlignment="Left" Height="36" VerticalAlignment="Top" Width="36" Stroke="#FF00A8FF"/>

<Image x:Name="User\_Profile\_256\_png" Margin="8,8,0,8" Source="User-Profile- 256.png" Stretch="Fill" RenderTransformOrigin="0.5,0.5" Width="20" Height="20" HorizontalAlignment="Left">

<Image.RenderTransform>

<TransformGroup>

<ScaleTransform ScaleX="-1"/>

<SkewTransform/>

<RotateTransform/>

<TranslateTransform/>

</TransformGroup>

</Image.RenderTransform>

</Image>

<Label x:Name="label5" Content="Ожидание игрока..." HorizontalAlignment="Left" Margin="40,13,0,0" VerticalAlignment="Top" Padding="0,0,0,-1" FontSize="8" Height="12" Width="75"/>

</Grid>

</Grid>

<Grid x:Name="loginframe" Background="#7FD0D0D0" Margin="0,0,0.333,0" Visibility="Collapsed">

<Label x:Name="portLbl" Content="Ваш порт:" HorizontalAlignment="Left"

Margin="10,480,0,0" VerticalAlignment="Top" Width="161"/>

<Grid HorizontalAlignment="Center" Height="219" Margin="0" VerticalAlignment="Center" Width="236" Background="#7FFFFFFF">

<Button x:Name="button1" Content="Играть" HorizontalAlignment="Left" Margin="40,145,0,0" VerticalAlignment="Top" Width="165" Height="38" Click="button1\_Click" IsDefault="True" Background="#00DDDDDD" BorderBrush="#FF008AFF"/>

<Rectangle HorizontalAlignment="Left" Height="34" VerticalAlignment="Top" Width="236" Fill="#FF00A8FF"/>

<Label x:Name="label" Content="Авторизация" HorizontalAlignment="Left" VerticalAlignment="Top" Background="{x:Null}" Foreground="White" Height="34" Width="236" HorizontalContentAlignment="Center" VerticalContentAlignment="Center"/>

<TextBox x:Name="loginedit" HorizontalAlignment="Left" Height="23" Margin="58,84,0,0" TextWrapping="Wrap" Text="Гость" VerticalAlignment="Top" Width="130"/>

<Label x:Name="label1" Content="Введите ваш ник:" HorizontalAlignment="Left" Margin="43,53,0,0" VerticalAlignment="Top" Height="26" RenderTransformOrigin="0.5,0.5" Width="165"/>

</Grid>

</Grid>

</Grid>

</Window>

Листинг A.3. – Листинг модуля NetworkAPI

using System;

using System.Runtime.InteropServices; using System.Text;

using System.Threading.Tasks; using static Kursach.pinvoke;

namespace Kursach

{

public class NetworkAPI

{

public class Packet

{

public string action;

public object value;

}

public class TCPServer

{

int s, s1;

public TCPServer(int port, Action<byte[]> RecieveCallback, Action<TCPServer> ConnectedCallback)

{

WSAData data = new WSAData();

int result = 0;

result = WSAStartup(WORD\_VERSION, ref data);

if (result == SUCCESS)

{

Console.WriteLine(data.szDescription);

s = (int)socket(ADDRESS\_FAMILIES.AF\_INET, SOCKET\_TYPE.SOCK\_STREAM, PROTOCOL.IPPROTO\_IP);

if (s < 0) throw new Exception("WinSock: Error calling socket");

sockaddr\_in addr = new sockaddr\_in();

addr.sin\_family = ADDRESS\_FAMILIES.AF\_INET;

addr.sin\_port = htons((ushort)port);

addr.sin\_addr = new in\_addr();

addr.sin\_addr.s\_addr = htonl(0x00000000);

if (Convert.ToBoolean(bind(new IntPtr(s), ref addr, Marshal.SizeOf(addr)))) throw new Exception("WinSock: Error calling bind");

if (Convert.ToBoolean(listen(new IntPtr(s), 10))) throw new Exception("WinSock: Error calling listen");

new Task(() => { waitConnection(new IntPtr(s), RecieveCallback, ConnectedCallback); }).Start();

}

}

public void stop(){

closesocket(new IntPtr(s));

WSACleanup();

Console.WriteLine("Destroyed");

}

void waitConnection(IntPtr socket, Action<byte[]> RecieveCallback, Action<TCPServer> ConnectedCallback){

s1 = (int)accept(new IntPtr(s), new IntPtr(0), 0);

if (s1 < 0) throw new Exception("WinSock: Error calling accept");

new Task(() => ConnectedCallback(this)).Start();

new Task(() => waitMessages(new IntPtr(s1), RecieveCallback)).Start();

}

void waitMessages(IntPtr socket, Action<byte[]> RecieveCallback){

byte[] buffer = new byte[256];

while (true) if (recv(socket, buffer, 256, 0) > 0) RecieveCallback(buffer);

}

public bool sendMessage(string text) => send(new IntPtr(s1), Encoding.UTF32.GetBytes(text), Encoding.UTF32.GetBytes(text).Length, 0) == new StringBuilder(text).Length;

}

public class TCPClient

{

int s;

public TCPClient(string ip, int port, Action<byte[]> RecieveCallback)

{

WSAData data = new WSAData();

int result = 0;

result = WSAStartup(WORD\_VERSION, ref data);

if (result == SUCCESS)

{

Console.WriteLine(data.szDescription);

s = (int)socket(ADDRESS\_FAMILIES.AF\_INET,

SOCKET\_TYPE.SOCK\_STREAM, PROTOCOL.IPPROTO\_IP);

if (s < 0) throw new Exception("WinSock: Error calling socket");

SockAddrIn addr = new SockAddrIn();

addr.Family = (ushort)ADDRESS\_FAMILIES.AF\_INET;

addr.Port = htons((ushort)port);

addr.Addr = inet\_addr(ip);

if (connect(new IntPtr(s), ref addr, Marshal.SizeOf(addr)) != 0)

{

throw new Exception("WinSock: Connection with the server failed...\n");

}

else

Console.WriteLine("Connected to the server...\n");

new Task(() => waitMessages(new IntPtr(s), RecieveCallback)).Start();

}

}

void waitMessages(IntPtr socket, Action<byte[]> RecieveCallback)

{

byte[] buffer = new byte[256];

while (true) if (recv(socket, buffer, 256, 0) > 0) RecieveCallback(buffer);

}

public void stop()

{

closesocket(new IntPtr(s));

WSACleanup();

}

public bool sendMessage(string text) => send(new IntPtr(s), Encoding.UTF32.GetBytes(text), Encoding.UTF32.GetBytes(text).Length, 0) == new StringBuilder(text).Length;

}

}

}

## Листинг A.4. – Листинг модуля PInvoke

using System;

using System.Collections.Generic; using System.Linq;

using System.Runtime.InteropServices; using System.Text;

using System.Threading.Tasks; using static Kursach.pinvoke;

namespace Kursach

{

public static class pinvoke

{

[DllImport("ws2\_32.dll", CharSet = CharSet.Auto, SetLastError = true)]

public static extern Int32 WSAStartup(Int16 wVersionRequested, ref WSAData wsaData);

[DllImport("ws2\_32.dll", CharSet = CharSet.Auto, SetLastError = true)]

public static extern Int32 WSACleanup();

[DllImport("ws2\_32.dll", CharSet = CharSet.Unicode, SetLastError = true)]

public static extern IntPtr socket(ADDRESS\_FAMILIES af, SOCKET\_TYPE socket\_type, PROTOCOL protocol);

[DllImport("Ws2\_32.dll", CharSet = CharSet.Unicode, EntryPoint = "WSAAddressToStringW")]

public static extern uint WSAAddressToString(ref sockaddr\_in lpsaAddress, int dwAddressLength, IntPtr lpProtocolInfo, StringBuilder lpszAddressString, ref int lpdwAddressStringLength);

[DllImport("Ws2\_32.dll")]

public static extern int bind(IntPtr s, ref sockaddr\_in addr, int addrsize);

[DllImport("Ws2\_32.dll")]

public static extern int listen(IntPtr s, int backlog);

[DllImport("Ws2\_32.dll")]

public static extern IntPtr accept(IntPtr s, IntPtr addr, int addrsize);

[DllImport("ws2\_32.dll")]

public static extern int recv(IntPtr Socket, [Out] StringBuilder buf, int len, int flags);

[DllImport("ws2\_32.dll")]

public static extern int recv(IntPtr Socket, [Out] byte[] buf, int len, int flags);

[DllImport("Ws2\_32.dll")]

public static extern int connect(IntPtr s, ref SockAddrIn addr, int addrsize);

[DllImport("ws2\_32.dll")]

public static extern int send(IntPtr Socket, [In] StringBuilder buf, int len, int flags);

[DllImport("ws2\_32.dll")]

public static extern int send(IntPtr Socket, [In] byte[] buf, int len, int flags);

[DllImport("ws2\_32.dll", CharSet = CharSet.Unicode, SetLastError = true)]

public static extern int closesocket(IntPtr s);

[StructLayout(LayoutKind.Sequential)]

public struct SockAddr

{

public ushort Family;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 14)]

public byte[] Data;

};

[StructLayout(LayoutKind.Sequential)]

public struct SockAddrIn

{

public ushort Family;

public ushort Port;

public uint Addr;

[MarshalAs(UnmanagedType.ByValArray, SizeConst = 8)]

public byte[] Zero;

}

[DllImport("Ws2\_32.dll", CharSet = CharSet.Unicode, EntryPoint = "WSAStringToAddressW")]

public static extern uint WSAStringToAddress(

string AddressString,

ADDRESS\_FAMILIES AddressFamily,

IntPtr lpProtocolInfo,

ref sockaddr\_in pAddr,

ref int lpAddressLength);

[DllImport("Ws2\_32.dll")]

public static extern ushort htonl(uint hostlong);

[DllImport("Ws2\_32.dll")]

public static extern ushort htons(ushort hostshort);

[DllImport("Ws2\_32.dll", CharSet = CharSet.Ansi)]

public static extern uint inet\_addr(string cp);

public enum ADDRESS\_FAMILIES : short

{

AF\_UNSPEC = 0, AF\_UNIX = 1, AF\_INET = 2, AF\_IMPLINK = 3, AF\_PUP = 4,

AF\_CHAOS = 5, AF\_NS = 6, AF\_IPX = 6, AF\_ISO = 7, AF\_OSI = 7, AF\_ECMA = 8,

AF\_DATAKIT= 9, AF\_CCITT = 10, AF\_SNA = 11, AF\_DECnet = 12, AF\_DLI = 13,

AF\_LAT= 14, AF\_HYLINK= 15, AF\_APPLETALK = 16, AF\_NETBIOS = 17,

AF\_VOICEVIEW = 18, AF\_FIREFOX = 19, AF\_UNKNOWN1 = 20, AF\_BAN = 21,

AF\_ATM = 22, AF\_INET6 = 23, AF\_CLUSTER = 24, AF\_12844 = 25, AF\_IRDA = 26,

AF\_NETDES = 28, AF\_TCNPROCESS = 29, AF\_TCNMESSAGE = 30, AF\_ICLFXBM = 31

}

public enum SOCKET\_TYPE : short

{

SOCK\_STREAM = 1,SOCK\_DGRAM = 2,SOCK\_RAW = 3,SOCK\_RDM = 4,SOCK\_SEQPACKET = 5

}

public enum PROTOCOL : short

{

IPPROTO\_IP = 0, IPPROTO\_ICMP = 1, IPPROTO\_IGMP = 2, IPPROTO\_GGP = 3,

IPPROTO\_TCP = 6, IPPROTO\_PUP = 12, IPPROTO\_UDP = 17, IPPROTO\_IDP = 22,

IPPROTO\_IPV6 = 41, IPPROTO\_ND = 77,SOCK\_SEQPACKET = 5

}

[StructLayout(LayoutKind.Sequential)]

public struct WSAData

{

const int WSADESCRIPTION\_LEN = 256;

const int WSASYS\_STATUS\_LEN = 128;

public Int16 wVersion;

public Int16 wHighVersion;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = WSADESCRIPTION\_LEN + 1)]

public String szDescription;

[MarshalAs(UnmanagedType.ByValTStr, SizeConst = WSASYS\_STATUS\_LEN + 1)]

public String szSystemStatus;

public Int16 iMaxSockets;

public Int16 iMaxUdpDg;

public IntPtr lpVendorInfo;

}

public const int SUCCESS = 0;

public const int HIGH\_VERSION = 2;

public const int LOW\_VERSION = 2;

public const short WORD\_VERSION = HIGH\_VERSION \* 256 + LOW\_VERSION;

}

[StructLayout(LayoutKind.Explicit, Size = 4)]

public struct in\_addr

{

[FieldOffset(0)] internal byte s\_b1;

[FieldOffset(1)] internal byte s\_b2;

[FieldOffset(2)] internal byte s\_b3;

[FieldOffset(3)] internal byte s\_b4;

[FieldOffset(0)] internal ushort s\_w1;

[FieldOffset(2)] internal ushort s\_w2;

[FieldOffset(0)] internal uint S\_addr;

public uint s\_addr

{

get { return S\_addr; }

set { S\_addr = s\_addr; }

}

public byte s\_host { get { return s\_b2; } }

public byte s\_net { get { return s\_b1; } }

public ushort s\_imp{ get { return s\_w2; } }

public byte s\_impno{ get { return s\_b4; } }

public byte s\_lh { get { return s\_b3; } }

}

[StructLayout(LayoutKind.Explicit, Size = 28)]

public struct sockaddr\_in

{

[FieldOffset(0)] internal ADDRESS\_FAMILIES sin\_family;

[FieldOffset(2)] internal ushort sin\_port;

[FieldOffset(4)] internal uint sin\_flowinfo;

[FieldOffset(8)] public in\_addr sin\_addr;

[FieldOffset(24)] internal uint sin\_scope\_id;

}

}