Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: «Компьютерные системы и сети»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему:

**Сетевая игра "Шашки"**

БГУИР КР 6 - 05 - 06 12 01 029 ПЗ

Студент Воскресенский Г.С.

Руководитель Болтак С.В.

Минск 2025

Учреждение образования  
«Белорусский государственный университет информатики  
 и радиоэлектроники»  
 УТВЕРЖДАЮ  
 Заведующий кафедрой ПОИТ   
  *–––––––––––* (подпись)  
  *–––––––*2025 г.

ЗАДАНИЕ  
по курсовому проектированию  
Студенту    Воскреснскому Георгию Сергеевичу–––––––––––––––––––

1. Тема работы   Сетевая игра «Шашки»––––––––––––––––––  
2. Срок сдачи студентом законченной работы––27.05.2025 г.–––     
3. Исходные данные к работе   *Программа реализует функционал пользователя. Язык программирования С#, среда разработки Microsoft Visual Studio. Вид приложения –Windows Forms, парадигма программирования – ООП. Способ организации данных – классы, способ хранения данных – файлы. Оформление кода должно соответствовать «Robert Martin Clear Code». Текст пояснительной записки оформляется по стандарту СТП 01–2024*  
4. Содержание расчётно-пояснительной записки (перечень вопросов, которые подлежат разработке)  
*Титульный лист––––––––––––––––––––––––––––––––––––           –– ––*  
*Задание по курсовой работе–––––––––– ––––––––––––––––           ––––*  
*Содержание ––––––––––––––––\_\_\_\_\_\_\_\_\_\_\_\_\_––––––––––           ––––*   
*Введение ––––––––––––––––\_\_\_\_\_\_\_\_\_\_\_\_\_––––––––––            –––––*  
*1 Анализ предметной области––––––––––––––––––––––– ––––––––*  
*1.1 Обзор аналогов ––– ––––––            –––вммкмкмкмкккииииии*  
*1.2 Постановка задачи–––– ––––– –             –––*  
*2 Проектирование программного средства––––––––––––––––––––––––*  
*2.1 Структура программы ––––––––– ––––––––*  
*2.2 Проектирование интерфейса программного средства –––––––––*  
*2.3 Проектирование функционала программного средства    – ––*   
*3 Разработка программного средства –\_\_\_\_\_\_\_\_\_\_\_\_\_.*   
*3.1* *Выбор технологий и инструментов\_ \_   ––––\_\_\_\_\_\_\_\_\_\_\_ \_\_\_. \_\_*  
*3.2 Реализация игровой логики ( .п.*   
*3.3 Работа с сокетами и сетью ( .*   
*3.4 Обработка пользовательского интерфейса*   
*4 Тестирование программного средства ––––––           –––– ––––––– 5* *Руководство пользователя \_\_   ––––\_\_\_\_\_\_\_\_\_\_. \_\_\_\_\_\_\_\_\_\_\_*  
*Заключение  \_\_\_\_\_\_\_\_\_\_\_\_\_\_. \_\_\_\_\_\_\_\_\_\_\_\_\_\_*  
*Список используемых источников*   
*Приложение А: Листинг кода с комментариями––––––– –––––––––––*

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

1. Схема алгоритма работы системы

6. Консультант по курсовой работе

Болтак С.В.

7. Дата выдачи задания 15.02.2025 г.–––––––––––––––––––––––   –

8. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и процентом от общего объёма работы):

раздел 1,2 к 15.03.2025 – 15 % готовности работы;

разделы 3, 4 к 15.03.2025 – 30 % готовности работы;

разделы 5, 6 к 15.03.2025 – 60 % готовности работы;

раздел 7, 8, 9 к 15.03.2025 – 90 % готовности работы;

оформление пояснительной записки и графического материала к 27.05.2025 – 100 % готовности работы.

Защита курсового проекта с 28.05.2025 по 7.06.2025 г.–––––––––––––––––––––

РУКОВОДИТЕЛЬ–––––– С.В.Болтак

(подпись)

Задание принял к исполнению ––Воскресенский Георгий Сергеевич    15.02.2025 г.

**СОДЕРЖАНИЕ**

Введение.........................................................................................................5

1 Анализ предметной области......................................................................7

1.1 Обзор аналогов........................................................................................7

1.2 Постановка задачи...................................................................................8

2 Проектирование программного средства.................................................11

2.1 Структура программы.............................................................................11

2.2 Проектирование интерфейса программного средства.........................12

2.3 Проектирование функционала программного средства....................13

3 Разработка программного средства........................................................14

3.1 Выбор технологий и инструментов.....................................................14

3.2 Реализация игровой логики..................................................................14

3.3 Работа с сокетами и сетью....................................................................15

3.4 Обработка пользовательского интерфейса.........................................15

4 Тестирование программного средства...................................................17

5 Руководство пользователя.......................................................................19

Заключение..................................................................................................23

Список использованных источников........................................................24

Приложение А. Исходный код программы..............................................25

# ВВЕДЕНИЕ

В последние годы компьютерные игры играют всё более важную роль не только в индустрии развлечений, но и как инструмент обучения программированию и разработки пользовательских интерфейсов. Среди настольных игр, перенесённых в цифровой формат, шашки занимают особое место благодаря своей простой логике и богатым возможностям для реализации алгоритмов искусственного интеллекта, взаимодействия между игроками и графического отображения.

Цель данного курсового проекта — разработка многопользовательской версии классической игры «Шашки» на языке программирования C# с использованием технологии Windows Forms. Программа должна поддерживать игру вдвоём за одним компьютером и по сети, отображать все основные элементы игрового процесса, обеспечивать корректную обработку ходов, обязательных взятий, правил движения и превращения шашек в дамки.

Основные задачи проекта:

1. Проанализировать правила и особенности реализации компьютерных версий игры «Шашки».
2. Разработать архитектуру приложения с поддержкой как локального, так и сетевого режима игры.
3. Реализовать графический интерфейс с использованием Windows Forms: визуализацию доски, шашек, дамок, кнопок управления и экранов окончания игры.
4. Обеспечить корректную реализацию логики игры: проверка допустимых ходов, реализация обязательного взятия, многократных ударов и перехода хода между игроками.
5. Реализовать сетевое взаимодействие с помощью TCP-сокетов: соединение между игроками, обмен информацией о ходе, окончании партии и выходе.
6. Протестировать работу приложения: стабильность соединения, корректность логики игры, работоспособность интерфейса и возврат в главное меню.

Практическая значимость проекта заключается в углублении навыков объектно-ориентированного программирования на C#, освоении работы с графическим интерфейсом Windows Forms и основами сетевого взаимодействия через сокеты, а также в создании полнофункциональной настольной игры, способной объединить двух игроков в одном или разных компьютерах.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

* 1. **Обзор аналогов**

Сетевые реализации игры «Шашки» получили широкое распространение как в учебных и развлекательных целях. Существуют различные варианты, отличающиеся по сложности, интерфейсу и технической реализации.

1. Веб-приложения, реализованные с использованием HTML5, JavaScript и WebSocket. Подобные версии позволяют играть онлайн через браузер, но требуют постоянного интернет-соединения и не всегда обеспечивают стабильную работу. Один из примеров – сайт 247checkers.com, представленный на рисунке 1.1.

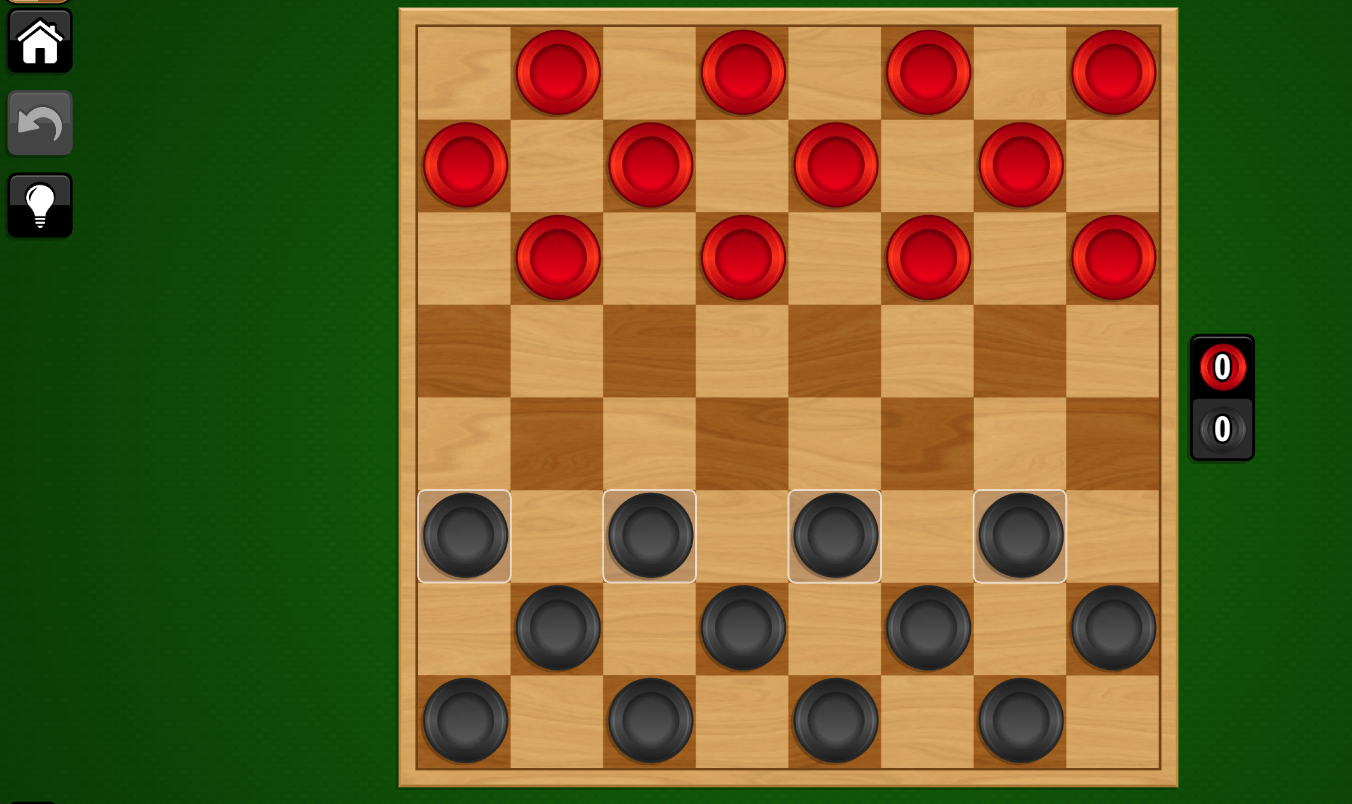


Рисунок 1.1 – сайт247checkers.com

2. Локальные Java-приложения, использующие TCP-сокеты для сетевого взаимодействия. Такие приложения демонстрируют основы клиент-серверного обмена, однако зачастую имеют упрощённый интерфейс и ограниченный функционал.

3. Открытые проекты на Python и Pygame, в которых реализована игра «Шашки» в однопользовательском режиме против ИИ или с базовой сетевой функциональностью без поддержки рематча, обработки исключений или проверки корректности сетевого обмена.

Разрабатываемое в рамках курсового проекта приложение отличается следующими особенностями:

- Использование языка программирования C# и технологии Windows Forms, что обеспечивает наглядный интерфейс и простоту интеграции с операционной системой Windows;

- Поддержка локального и сетевого режимов игры: один игрок создаёт сервер, другой подключается по IP-адресу;

- Полная игровая логика: обязательные взятия, серийные удары, превращение в дамку, контроль очереди хода и победных условий;

- Функции управления партией: кнопки «Сдаться» и «Предложить ничью» с учётом сетевого обмена;

- Простота запуска и низкие системные требования: достаточно Windows с установленной .NET-платформой.

**1.2 Постановка задачи**

Цель проекта — создать сетевую версию игры «Шашки» на языке C# с использованием Windows Forms и TCP-сокетов.

Основные требования:

1. Главное меню с кнопками «Создать сервер», «Присоединиться», «Играть» и «Выйти».

- При создании сервера отображается IP-адрес и надпись «Ожидаем соперника», сервер ждёт подключения с таймаутом 10 секунд.

- При выборе «Присоединиться» вводится IP сервера, выполняется попытка соединения с таймаутом 5 секунд.

- При ошибке (порт занят, отказ подключения) выводится соответствующее сообщение об ошибке.

2. После установления соединения отображается игровая доска 8×8 (размер клетки 60×60, общая площадь 600×600 пикселей).

- Шашки черного цвета размещаются на первых трёх рядах, белого — на последних трёх рядах.

- Сервер играет за белых (ходит первым), клиент — за чёрных.

3. Обработка кликов по клеткам:

- При клике по своей шашке выделяются допустимые ходы (зеленым цветом) и возможные захваты (красным цветом).

- При перемещении шашки отправляется сообщение формата “MOVE:fromX,fromY:toX,toY” по сети.

- При получении сообщения от соперника ход воспроизводится локально, выполняется проверка победы.

4. Логика ходов и захватов:

- Обычная шашка ходит по диагонали вперёд на одну клетку.

- Дамка (король) перемещается по диагонали на любое количество свободных клеток.

- Обязательный захват: если доступен хоть один захват, простые ходы запрещены.

- Многоходовые серии: после каждого захвата проверяется возможность нового захвата той же шашкой; при наличии доступных захватов игрок продолжает ход своей шашкой.

- При захвате шашка противника удаляется с доски.

5. Преобразование в дамку:

- Белая шашка, достигшая верхнего ряда (X=0), и чёрная, достигшая нижнего ряда (X=7), становятся дамками.

- Визуально дамка обозначается короной (символ “♛”).

6. Подсветка возможных ходов:

- Клетки, доступные для простого хода, подсвечиваются светло-зелёным; клетки, в которые можно выполнить захват, подсвечиваются оранжево-красным.

7. Окончание игры:

- Если у игрока нет возможных ходов или захватов, он проигрывает; ничья возможна только по согласию обоих игроков.

- Кнопка «Сдаться» завершает игру с победой соперника; в сетевом режиме отправляется команда «SURRENDER».

- Кнопка «Предложить ничью» отправляет команду «DRAW\_REQUEST», при согласии соперника отправляется «DRAW\_ACCEPT» и игра завершается ничьей.

- После завершения игры выводятся экраны «Победа белых!», «Победа чёрных!» или «Ничья!» с кнопкой «В главное меню».

8. Интерфейс:

- Главное окно размером 600×650 пикселей: 600×600 для доски и 50×650 для кнопок («Сдаться», «Предложить ничью», «В главное меню»).

- Используются стандартные элементы Windows Forms (Panel, Button, Label).

**2 ПРОЕКТИРОВАНИЕ ЗАДАЧИ**

**2.1 Структура программы**

Разработанная программа представляет собой сетевую реализацию классической настольной игры «Шашки», выполненную на языке программирования C# с использованием технологии Windows Forms. Игровое взаимодействие между двумя игроками реализуется через TCP-протокол с использованием клиент-серверной архитектуры. Программа состоит из нескольких ключевых компонентов, каждый из которых отвечает за отдельную часть логики или интерфейса:

- Сетевой модуль (Server, Client, NetworkManager): реализует установку соединения между игроками, обмен сообщениями о ходах, предложениях ничьей, сдаче и завершении игры. Передача данных осуществляется в строковом формате с простейшей сериализацией и последующей обработкой на стороне клиента и сервера.

- Графический интерфейс (Form1, GameForm, JoinServerForm, CreateServerForm): реализует визуальную часть программы, включая главное меню, подключение к серверу, отображение IP-адреса, игровую доску, кнопки управления (сдаться, ничья, в меню) и экраны завершения партии. Интерфейс выполнен с использованием стандартных компонентов Windows Forms (Panel, Button, Label) и оформлен в едином стиле.

- Логика игры (GameForm.cs): реализует основную механику игры – расстановка шашек, определение возможных ходов и захватов, обязательное взятие, превращение в дамку, серийные удары, переход хода и завершение игры. Все действия пользователя и события, полученные по сети, синхронизируются и визуализируются в реальном времени.

- Компонент CheckerPiece.cs: представляет отдельную шашку, визуализированную в виде круглого элемента. Поддерживает свойство IsKing для обозначения дамки, а также отрисовку короны при достижении последней линии.

Благодаря модульной структуре обеспечивается удобство расширения и тестирования проекта, а логика сетевого взаимодействия отделена от пользовательского интерфейса для стабильной и предсказуемой работы приложения.

**2.2 Логическое моделирование**

Разрабатываемая сетевая игра «Шашки» предназначена для двух игроков, подключающихся по локальной сети. Оба игрока играют по очереди, видят текущую расстановку фигур и могут взаимодействовать через игру (сдача, ничья, ход).

Основными логическими компонентами системы являются:

- Игрок – участник сессии, играющий за белых или чёрных. Содержит информацию о цвете, очередности хода, статусе подключения и результатах игры.

- Шашка – графический и логический объект, размещаемый на игровом поле. Содержит данные о позиции, цвете, статусе (обычная/дамка).

- Игровое поле – матрица 8×8, представленная массивом панелей. Каждая клетка хранит информацию о наличии/отсутствии шашки и доступности хода.

- Сервер – узел, ожидающий подключения клиента. После установления соединения отправляет и получает сообщения, маршрутизирует данные между игроками.

- Клиент – подключается по IP-адресу к серверу, участвует в игровом процессе на равных правах. Получает и обрабатывает игровые события, отображает доску и действия противника.

Логика взаимодействия:

1. Один игрок запускает сервер и ожидает подключения.

2. Второй игрок подключается, и устанавливается соединение.

3. Сервер назначает цвета: белые – сервер, чёрные – клиент.

4. Игроки по очереди делают ходы, взаимодействие синхронизируется через сетевые сообщения.

5. В случае захвата выполняется проверка возможности серии ударов.

6. При отсутствии ходов или сдаче одного из игроков – игра завершается, оба игрока получают сообщение о результате.

Таким образом, логическая модель построена на централизованной передаче команд, с разделением ролей между клиентом и сервером, обеспечивая корректное поведение и интерактивность игрового процесса.

**2.3 Выбор и обоснование инструментов разработки**

Для реализации сетевой игры «Шашки» выбран стек технологий, позволяющий обеспечить устойчивую работу в сетевом и локальном режимах, интуитивно понятный интерфейс и стабильную передачу данных:

- Язык программирования: C#

• Высокая скорость выполнения и поддержка объектно-ориентированного подхода.

• Широкие возможности для работы с сетями и графическим интерфейсом.

- Платформа: .NET Framework

• Использование встроенных библиотек для TCP-соединений (TcpListener, TcpClient).

• Поддержка событийной модели и многопоточности.

- Интерфейс: Windows Forms

• Простая реализация визуальных элементов.

• Возможность быстрого отображения и обновления состояния доски и фигур.

- Формат обмена данными: текстовые строки (команды вида «MOVE:x1,y1:x2,y2», «DRAW\_REQUEST», «SURRENDER»).

Выбранные инструменты полностью соответствуют поставленным целям и обеспечивают надёжную реализацию игрового процесса с минимальными зависимостями и высоким уровнем управляемости.

**3 РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**3.1 Выбор технологий и инструментов**

Для разработки сетевой игры «Шашки» использовались следующие инструменты и технологии:

- Язык программирования: C#.

- Графическая библиотека: Windows Forms (для создания интерфейса и отрисовки доски, шашек, дамок и кнопок).

- Сетевой модуль: TCP-сокеты через классы TcpClient и TcpListener из пространства имён System.Net.Sockets.

- Платформа: .NET Framework.

- Среда разработки: Microsoft Visual Studio Community Edition.

**3.2 Реализация игровой логики**

Вся игровая логика реализована в файле GameForm.cs. Основные этапы:

1. Инициализация доски:

- Создание сетки 8×8 с помощью массива Panel[], boardPanels.

- Расстановка начальных фигур: белые – внизу, чёрные – вверху.

- Отображение шашек и визуализация дамок с помощью символа «♛».

2. Переменные состояния:

- selectedPiece – текущая выбранная шашка;

- selectedPanel – ячейка с выбранной шашкой;

- currentPlayer – текущий игрок (white или black);

- mustContinueCapture – флаг, указывающий на необходимость продолжения взятия.

3. Правила игры:

- Проверка допустимых ходов (вперёд/назад, диагонали);

- Обязательное взятие, в том числе при возможности серийного удара;

- Преобразование в дамку при достижении последней линии;

- Удаление побитых шашек после хода.

4. Подсветка:

- Зеленым цветом – доступные ходы;

- Красно-оранжевым – возможные захваты.

5. Завершение партии:

- Проверка наличия возможных ходов у текущего игрока;

- Показ окна победы или ничьей.

**3.3 Работа с сокетами и сетью**

Сетевая часть реализована с помощью TCP-сокетов и включает серверную и клиентскую составляющие.

1. Сервер:

- Использует TcpListener для прослушивания соединений;

- Принятие клиента и создание потока обмена;

- Отправка данных о ходе и полученных событиях (ход, сдача, ничья, выход).

2. Клиент:

- Подключается к серверу через TcpClient;

- Принимает данные и выполняет соответствующие действия.

3. Сообщения:

- MOVE:x1,y1:x2,y2 — движение шашки;

- SURRENDER — сдача игрока;

- DRAW\_REQUEST — предложение ничьей;

- DRAW\_ACCEPT / DRAW\_REJECT — согласие/отказ от ничьей;

- EXIT — завершение соединения и выход в меню.

4. Поток приёма:

- Запускается параллельно с игрой;

- Получает сообщения и вызывает соответствующую обработку на стороне GameForm.

**3.4 Обработка пользовательского интерфейса**

Интерфейс реализован в Windows Forms и включает несколько ключевых компонентов:

1. Главное меню (Form1):

- Кнопки: «Создать сервер», «Присоединиться», «Играть», «Выйти».

2. Окна подключения (CreateServerForm, JoinServerForm):

- Отображают IP сервера, статус подключения, ошибки соединения.

3. Игровое окно (GameForm):

- Доска из 64 панелей (8×8);

- Отображение шашек и дамок;

- Метки с текстом «Ваш ход» или «Ход соперника»;

- Кнопки «Сдаться», «Предложить ничью», «В главное меню».

4. Окна окончания партии:

- Сообщения о победе, ничьей, поражении;

- Кнопка возврата в меню;

- Поддержка согласия на ничью только при подтверждении обоих игроков.

5. Обновление интерфейса:

- Выполняется через методы Invalidate() и ручную перерисовку панели;

- Используется ToggleSelect() для визуального выделения шашки.

**4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

Тестирование игры «Шашки» проводилось вручную и охватывало следующие ключевые этапы:

1. Проверка запуска сервера:

- При нажатии кнопки «Создать сервер», если порт свободен – сервер успешно запускается, отображается сообщение «Ожидаем соперника…».

- При повторном запуске или занятом порте – появляется сообщение «Не удалось создать сервер (уже запущен?)».

2. Проверка подключения клиента:

- При корректном вводе IP и активном сервере – клиент успешно подключается, начинается игра.

- При попытке подключения к несуществующему серверу — через 5 секунд выводится сообщение «Ошибка подключения: ...».

3. Проверка игрового процесса:

- Отображение доски, правильная расстановка шашек на начальных позициях.

- При нажатии на шашку подсвечиваются допустимые ходы.

- Перемещение шашки на свободную клетку и удаление побитой шашки у противника.

- После завершения хода – смена хода и корректная надпись: «Ваш ход» или «Ход соперника».

- В сетевом режиме: синхронное отображение ходов у обоих игроков.

4. Проверка правил победы и ничьей:

- Победа засчитывается, если у противника нет допустимых ходов — появляется окно с сообщением о победителе.

- Ничья возможна только по взаимному согласию: при предложении ничьей от одного игрока, второй должен подтвердить.

5. Проверка работы дамок:

- Шашка превращается в дамку при достижении противоположной линии.

- Дамка совершает ходы и взятия в любом направлении по диагонали.

- Реализовано взятие дамкой серией – при возможности продолжения серии ход не завершается.

6. Проверка сетевых сообщений:

- Передача и приём ходов, сдачи (SURRENDER), ничьей (DRAW\_REQUEST/ACCEPT/REJECT), выхода (EXIT) работает корректно.

- Отключение одного из игроков приводит ко всплывающему сообщению и возврату в главное меню.

7. Проверка выхода:

- При нажатии кнопки «В главное меню» – оба игрока корректно возвращаются в меню, соединение разрывается.

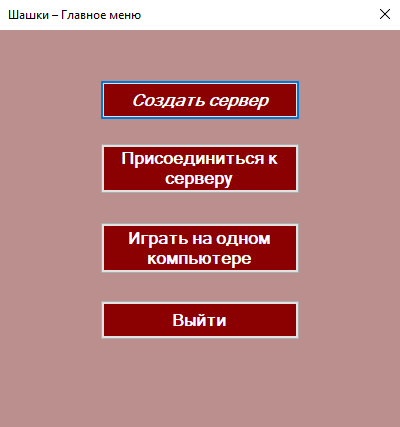
- Отправка команды «EXIT» подтверждается на обеих сторонах.

**5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

Программное средство «Шашки» – это сетевая игра для двух игроков, работающая в режиме реального времени. В данном разделе приведён пошаговый сценарий взаимодействия с программой со стороны пользователя.

**Шаг 1. Запуск программы**

При входе в программу пользователя встречает главное меню, где пользователь может выбрать дальнеёшее действие. Показано на рисунке 5.1.

  
Рисунок 5.1 – Главное меню

**Шаг 2. Создание сервера**

После того как пользователь выбирает создать сервер, программа создаёт окно для ожидания соперника, который должен подключиться по IP-адресу, который показан в окне. Работа формы показана на рисунке 5.2.

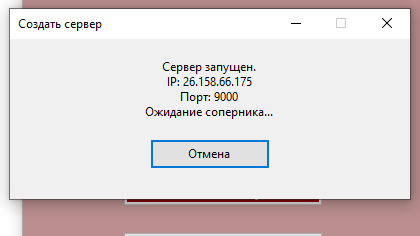


Рисунок 5.2 – Окно создания сервера

**Шаг 3. Подключение к серверу**

Второй пользователь в своём меню должен выбрать присоединиться к серверу, после чего в появившемся окне вписать IP-адрес сервера, к которому хочет подключиться. Пример показан на рисунке 5.3.

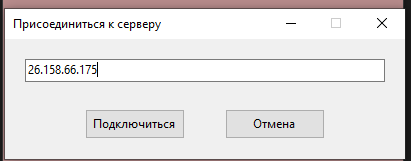


Рисунок 5.3 - Регистрация

**Шаг 4. Игра**

После успешного ввода IP-адреса пользователи попадают в игру, где играют по правилам русских шашек, имея возможность сдаться или предложить ничью.  
Игровое поле изображено на рисунке 5.4.

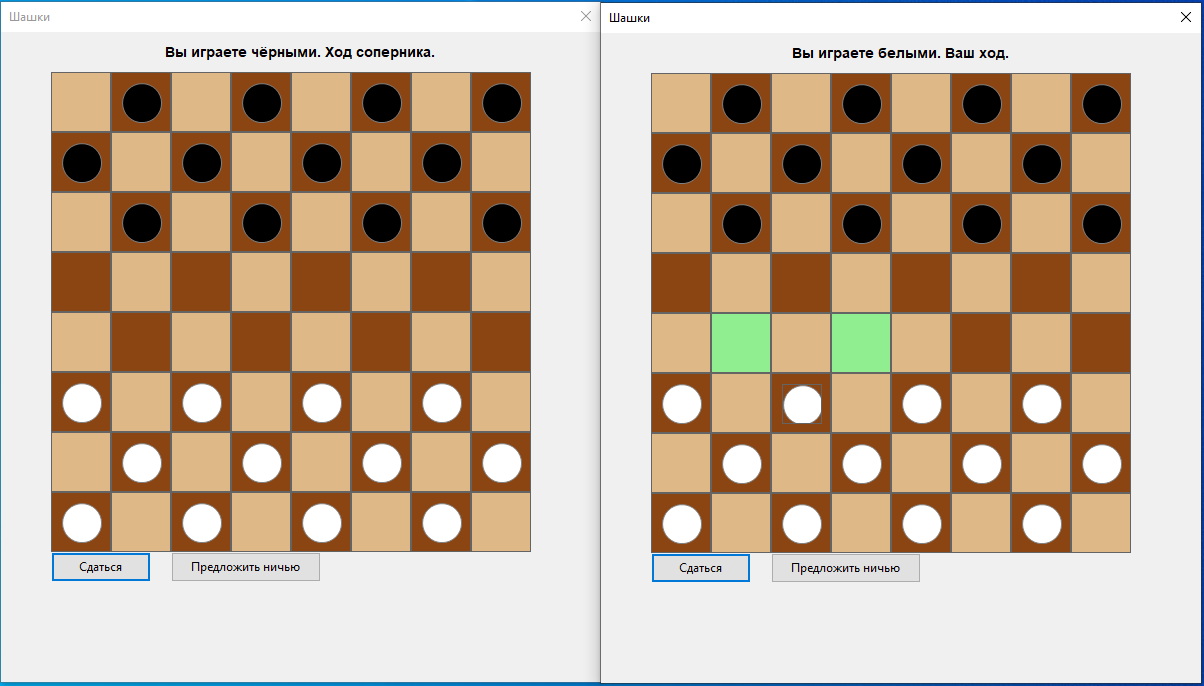
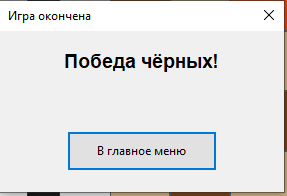


Рисунок 5.4 – Игровое поле

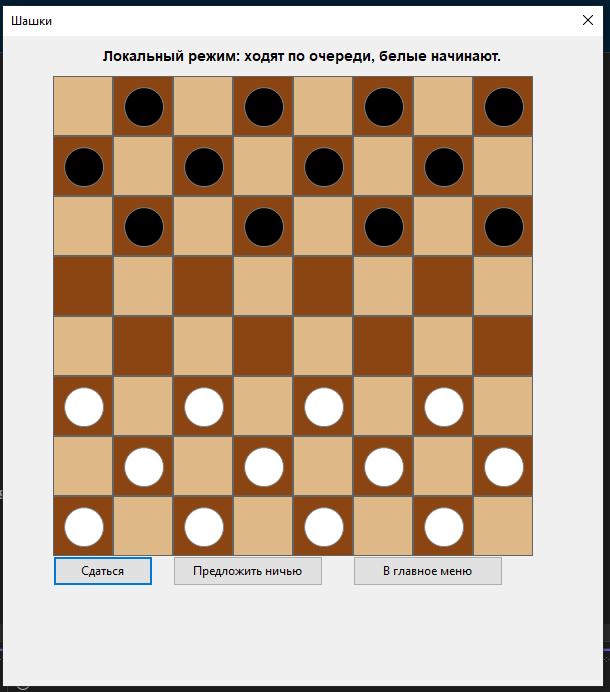
**Шаг 5. Окончание игры**

После завершения игры появляется окно, сообщающее о результатах и дающее возможность выхода в главное меню.  
Окончание игры изображено на рисунке 5.5.

  
  
Рисунок 6 – Окончание игры

**Шаг 6. Локальная игра**

Пользователь также имеет возможность запустить игру на двоих с одного компьютера по тем же правилам. Игровое окно локальной игры изображено на рисунке 5.6.

  
  
Рисунок 5.6 – Игровое окно локальной игры

**ЗАКЛЮЧЕНИЕ**

В рамках выполнения курсового проекта была разработана сетевая реализация игры «Шашки», включающая основные элементы классических правил, а также поддержку многопользовательского взаимодействия через локальную сеть. Приложение создано с использованием языка программирования C# и технологий Windows Forms и .NET, что позволило добиться высокой производительности и визуальной наглядности при сравнительно низких системных требованиях.

Разработанная игра обеспечивает два режима взаимодействия – локальный (за одним компьютером) и сетевой (через TCP-соединение). Реализована вся необходимая логика: расстановка шашек, перемещение, обязательные взятия, дамки с многоударными сериями, проверка на окончание партии, предложения ничьей и сдачи, а также интерфейс возврата в главное меню.

К числу преимуществ разработанного программного средства можно отнести:

- Интуитивно понятный графический интерфейс;

- Полная реализация правил шашек, включая дамок и обязательные взятия;

- Поддержка сетевой игры между двумя пользователями без использования внешних библиотек;

- Минимальные системные требования, что обеспечивает запуск на большинстве современных компьютеров.

К возможным недостаткам текущей версии можно отнести отсутствие полноценной рейтинговой системы, ограничение на использование в пределах локальной сети, а также отсутствие поддержки масштабирования и игры более чем между двумя игроками.

Разработка игры позволила закрепить навыки работы с графическим интерфейсом Windows Forms, сетевым взаимодействием через TCP-сокеты, а также реализацию сложной логики многократных ударов и победных условий. Созданное приложение демонстрирует потенциал к дальнейшему расширению: добавление интернет-сервера, онлайн-рейтинг, а также улучшение интерфейса и анимации.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] MSDN Library – Документация по C# и .NET Framework [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/, свободный.

[2] Microsoft Docs – TcpClient и TcpListener [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/api/system.net.sockets, свободный.

[3] Petzold C. Programming Windows Forms with C#. – Microsoft Press, 2005. – 550 с.

[4] Troelsen A. Pro C# 8 with .NET Core 3. – Apress, 2020. – 1080 с.

[5] Стукалов А.Ю. Основы объектно-ориентированного программирования на C#. – СПб.: Питер, 2021. – 384 с.

[6] Официальные правила русских шашек / Федерация шашек России [Электронный ресурс]. – Режим доступа: https://shashki.ru/rules/, свободный.

[7] Кравец С.Ю. Разработка сетевых приложений на C#. Учебное пособие. – М.: БХВ-Петербург, 2018. – 320 с.

[8] Stack Overflow – Сообщество разработчиков C# и .NET [Электронный ресурс]. – Режим доступа: https://stackoverflow.com/, свободный.

[9] GitHub – Примеры реализаций настольных игр на C# [Электронный ресурс]. – Режим доступа: https://github.com/, свободный.

# Приложение А: листинг кода с комментариями

using System;

using System.Windows.Forms;

namespace kurs

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void btnCreateServer\_Click(object sender, EventArgs e)

{

var serverForm = new CreateServerForm(); // форма с логикой сервера

serverForm.Show();

}

private void btnJoinServer\_Click(object sender, EventArgs e)

{

var connectForm = new JoinServerForm(); // форма ввода IP и подключения

connectForm.Show();

}

private void btnLocalPlay\_Click(object sender, EventArgs e)

{

var gameForm = new GameForm();

gameForm.Show();

this.Hide(); // скрываем главное меню

}

private void btnExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

}

}

using kurs;

using System.Drawing;

using System.Windows.Forms;

public class CheckerPiece : Panel

{

public bool IsKing { get; set; } = false;

public CheckerPiece(string color)

{

this.Width = 40;

this.Height = 40;

this.BackColor = Color.Transparent;

this.Tag = color;

this.Paint += CheckerPiece\_Paint;

}

private void CheckerPiece\_Paint(object sender, PaintEventArgs e)

{

Graphics g = e.Graphics;

Color pieceColor = (string)Tag == "white" ? Color.White : Color.Black;

Brush brush = new SolidBrush(pieceColor);

g.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

g.FillEllipse(brush, 0, 0, Width - 1, Height - 1);

g.DrawEllipse(Pens.Gray, 0, 0, Width - 1, Height - 1);

if (IsKing)

{

Font font = new Font("Arial", 14, FontStyle.Bold);

TextRenderer.DrawText(g, "♛", font, new Rectangle(0, 0, Width, Height),

pieceColor == Color.White ? Color.Black : Color.White,

TextFormatFlags.HorizontalCenter | TextFormatFlags.VerticalCenter);

}

}

public void ToggleSelect()

{

this.BorderStyle = this.BorderStyle == BorderStyle.None ? BorderStyle.FixedSingle : BorderStyle.None;

}

protected override void OnClick(EventArgs e)

{

base.OnClick(e);

if (this.Parent is Panel panel && this.FindForm() is GameForm form)

{

form.HandleCellClick(panel);

}

}

}

using System;

using System.Drawing;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace kurs

{

public partial class GameForm : Form

{

private const int BoardSize = 8;

private Panel[,] boardPanels = new Panel[BoardSize, BoardSize];

private CheckerPiece selectedPiece = null;

private Panel selectedPanel = null;

private string currentPlayer = "white";

private bool mustContinueCapture = false;

private readonly Color HighlightColor = Color.LightGreen;

private readonly Color CaptureHighlightColor = Color.OrangeRed;

private readonly Color DefaultCellColor1 = Color.BurlyWood;

private readonly Color DefaultCellColor2 = Color.SaddleBrown;

// Сетевой режим

private TcpClient netClient = null;

private NetworkStream netStream = null;

private bool isServer = false;

private bool myTurn = false;

private string myColor = null;

// UI

private Label lblStatus;

private Button btnSurrender;

private Button btnDraw;

private Button btnMenu; // Кнопка «В главное меню» (только для локального режима)

#region Конструкторы

public GameForm()

{

InitializeComponent();

InitializeBoard();

AddControlButtons(localMode: true);

lblStatus.Text = "Локальный режим: ходят по очереди, белые начинают.";

}

public GameForm(TcpClient client, bool isServer)

{

InitializeComponent();

InitializeBoard();

AddControlButtons(localMode: false);

this.netClient = client;

this.netStream = client.GetStream();

this.isServer = isServer;

myColor = isServer ? "white" : "black";

currentPlayer = "white";

myTurn = (myColor == "white");

UpdateStatusLabel();

Task.Run(() => ListenForMessages());

}

#endregion

#region Инициализация компонентов и доски

private void InitializeComponent()

{

this.lblStatus = new Label();

this.SuspendLayout();

//

// lblStatus

//

this.lblStatus.Location = new Point(50, 10);

this.lblStatus.Name = "lblStatus";

this.lblStatus.Size = new Size(500, 20);

this.lblStatus.TextAlign = ContentAlignment.MiddleCenter;

this.lblStatus.Font = new Font("Arial", 10, FontStyle.Bold);

//

// GameForm

//

this.ClientSize = new Size(600, 650);

this.Controls.Add(this.lblStatus);

this.FormBorderStyle = FormBorderStyle.FixedDialog;

this.MaximizeBox = false;

this.MinimizeBox = false;

this.Name = "GameForm";

this.StartPosition = FormStartPosition.CenterScreen;

this.Text = "Шашки";

this.ResumeLayout(false);

}

private void InitializeBoard()

{

int offsetX = 50, offsetY = 40;

int cellSize = 60;

for (int row = 0; row < BoardSize; row++)

{

for (int col = 0; col < BoardSize; col++)

{

Panel cell = new Panel

{

Size = new Size(cellSize, cellSize),

Location = new Point(offsetX + col \* cellSize, offsetY + row \* cellSize),

BackColor = (row + col) % 2 == 0 ? DefaultCellColor1 : DefaultCellColor2,

BorderStyle = BorderStyle.FixedSingle,

Tag = new Point(row, col)

};

cell.Click += Cell\_Click;

this.Controls.Add(cell);

boardPanels[row, col] = cell;

if ((row + col) % 2 != 0 && row < 3)

AddPiece(cell, "black");

else if ((row + col) % 2 != 0 && row > 4)

AddPiece(cell, "white");

}

}

}

private void AddPiece(Panel cell, string color)

{

var piece = new CheckerPiece(color);

piece.Location = new Point((cell.Width - piece.Width) / 2,

(cell.Height - piece.Height) / 2);

cell.Controls.Add(piece);

}

/// <summary>

/// Добавляет кнопки «Сдаться» и «Предложить ничью».

/// Если localMode == true, добавляем ещё и «В главное меню».

/// </summary>

private void AddControlButtons(bool localMode)

{

// Кнопка "Сдаться"

btnSurrender = new Button

{

Text = "Сдаться",

Location = new Point(50, 520),

Size = new Size(100, 30)

};

btnSurrender.Click += BtnSurrender\_Click;

this.Controls.Add(btnSurrender);

// Кнопка "Предложить ничью"

btnDraw = new Button

{

Text = "Предложить ничью",

Location = new Point(170, 520),

Size = new Size(150, 30)

};

btnDraw.Click += BtnDraw\_Click;

this.Controls.Add(btnDraw);

if (localMode)

{

// Кнопка "В главное меню" только для локального режима

btnMenu = new Button

{

Text = "В главное меню",

Location = new Point(350, 520),

Size = new Size(150, 30)

};

btnMenu.Click += (s, e) =>

{

this.Close();

new Form1().Show();

};

this.Controls.Add(btnMenu);

}

}

#endregion

#region Обработка клика по клетке

private void Cell\_Click(object sender, EventArgs e)

{

if (netClient != null && !myTurn)

return;

Panel targetCell = sender as Panel;

if (targetCell == null) return;

if (mustContinueCapture)

{

if (selectedPiece == null || selectedPanel == null) return;

Panel capturedPanel;

if (!IsValidMove(selectedPanel, targetCell, out capturedPanel)) return;

PerformMove(targetCell, capturedPanel);

return;

}

if (selectedPiece != null && selectedPanel != null)

{

Panel capPanel;

if (IsValidMove(selectedPanel, targetCell, out capPanel))

{

PerformMove(targetCell, capPanel);

return;

}

}

if (targetCell.Controls.Count > 0)

{

var piece = targetCell.Controls[0] as CheckerPiece;

if (netClient != null)

{

if (!myTurn) return;

if ((string)piece.Tag != myColor) return;

}

else

{

if ((string)piece.Tag != currentPlayer) return;

}

string checkColor = (netClient != null) ? myColor : currentPlayer;

if (HasAnyCapture(checkColor) && !HasCaptureMoves(targetCell))

return;

selectedPiece?.ToggleSelect();

selectedPiece = piece;

selectedPiece.ToggleSelect();

selectedPanel = targetCell;

HighlightValidMoves(targetCell);

}

}