Федеральное государственное автономное образовательное учреждение

высшего образования «Санкт-Петербургский государственный

электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)»

**Кафедра информационных систем**

**ОТЧЕТ**

**по курсовой работе**

**по дисциплине «Объектно-ориентированное программирование»**

Студент гр. №3376 Михайлов Н.  
Студентка гр. №3376 Дегтярева М.  
  
Преподаватель Егоров С.С.

Санкт-Петербург  
2024

1. Задание на курсовую работу:   
  
1.1 Описание предметной области

Два игрока каждый на своем поле NхN, N>3 (9х9) независимо друг от друга расставляют цифры от 1 до M (9). Нельзя заполнять клетки, смежные по горизонтали и вертикали, но возможно по диагонали. По команде «начало игры» сопернику предъявляется собственное закрытое поле. Игроки по очереди открывают 1 клетку соперника. Все ходы играющих отображаются на их полях. Игра оканчивается при открытии всех заполненных полей соперника или по выполнении K (50) ходов. Выиграл тот, сумма которого больше.

Дополнительные требования:

- игра сетевая, т.е. приложение отвечает только за 1 игрока,

- для каждого игрока приложение одно и то же,

- первым ход делает игрок, приложение которого запущено первым,

- возможна работа на одном компьютере.

1.2 Синтаксический анализ задания

Таблица синтаксического анализа

|  |  |
| --- | --- |
| **Лексема** | **Элемент объектной модели** |
| Игра(игрок) | Класс |
| Поле | Класс |
| Клетка | Класс |
| Размер поля | Характеристика поля |
| Цифра | Характеристика клетки |
| Ход | Метод игры(игрока) |
| Сумма значений | Характеристика игры(игрока) |
| Победа | Метод игры(игрока) |
| Сетевое взаимодействие | Класс |

«Меню» - класс, неявно выявленный при синтаксическом разборе.

2. Диаграммы объектной модели

2.1 Модель сущность-связь предметной области:

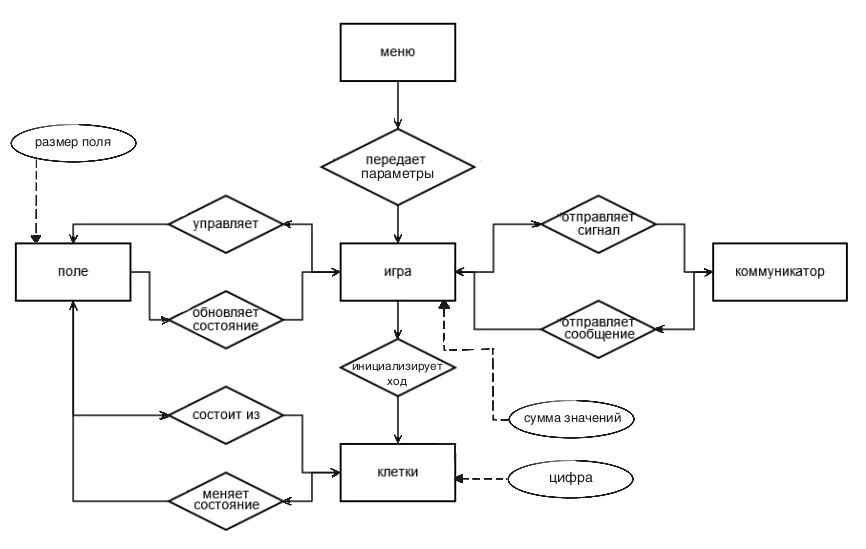


Рисунок 1 - модель сущность-связь

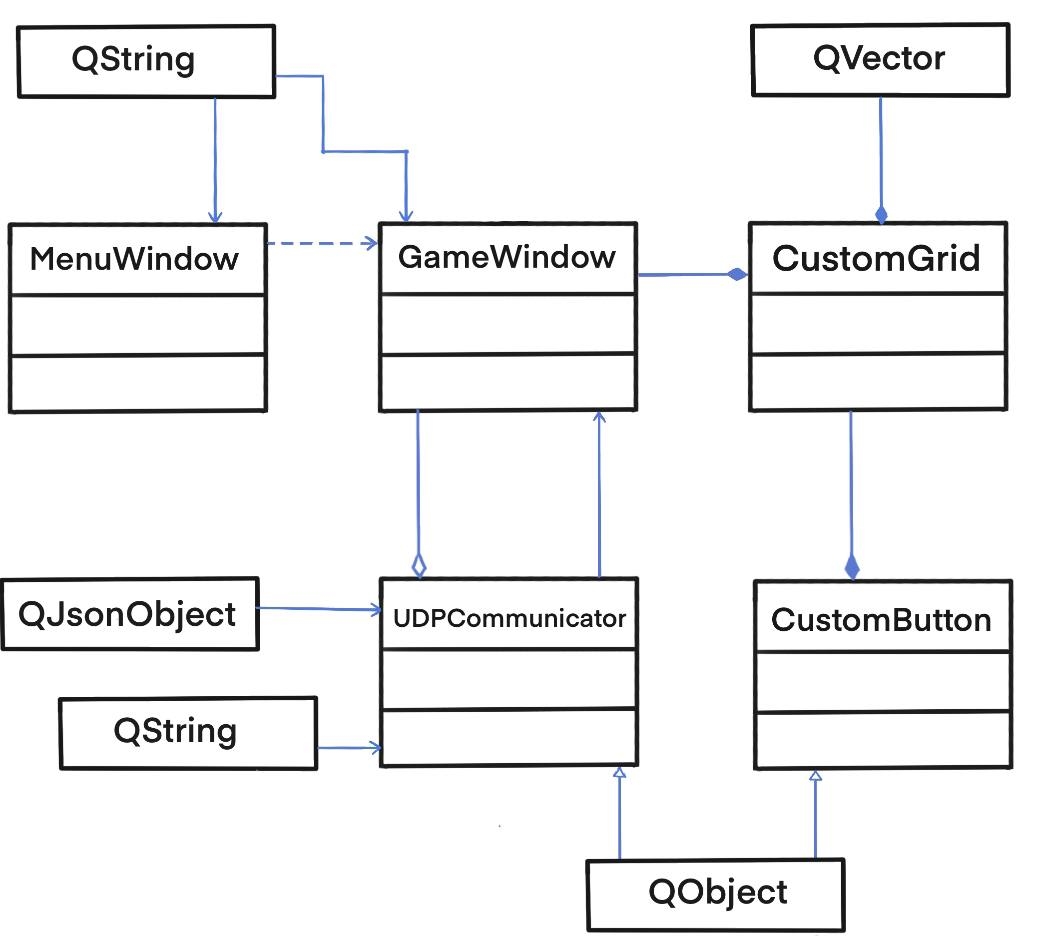
2.2 Диаграмма классов:  
  


Рисунок 2 – диаграмма классов

Таблица соответствия

|  |  |  |
| --- | --- | --- |
| **Предметная область** | **Модель «сущность-связь»** | **Диаграмма классов** |
| Игра(игрок) | Игра | GameWindow |
| Поле | Поле | CustomGrid |
| Клетки | Клетки | CustomButton |
| Сетевое взаимодействие | Коммуникатор | UDPCommunicator |
| Меню | Меню | MenuWindow |

Таблица атрибутов классов

|  |  |  |
| --- | --- | --- |
| **Наименование**  **атрибута** | **Идентификатор : тип** | **Назначение** |
| Класс: GameWindow | | |
| Текущий ход | turn : int | Отслеживание текущего хода в игре. |
| Создатель игры | isGameCreator : bool | Признак, является ли игрок создателем игры. |
| Имя игрока | nikname : QString | Хранение имени игрока |
| Название игры | gameName : QString | Хранение названия игры. |
| Поле игрока. | playerGrid : CustomGrid\* | Объект поля для игрока. |
| Поле соперника. | opponentGrid : CustomGrid\* | Объект поля для соперника. |
| Сетевой коммуникатор | communicator : UDPCommunicator\* | Объект для сетевого взаимодействия. |
| Класс: CustomGrid | | |
| Количество строк | rows : int | Количество строк на игровом поле. |
| Количество столбцов | cols : int | Количество столбцов на игровом поле. |
| Список кнопок | buttons : QVector<CustomButton\*> | Список кнопок на поле. |
| Количество правильных клеток | rightButtons : int | Количество правильных кнопок. |
| Сумма значений клеток | summ : int | Сумма значений клеток игрока. |
| Класс: CustomButton | | |
| Значение клетки | value : int | Значение клетки. |
| Номер строки | row : int | Номер строки клетки. |
| Номер столбца | col : int | Номер столбца клетки. |
| Проверено соперником | checkedByEnemy : bool | Проверена ли клетка соперником. |
| Проверено текущим игроком | alreadyChecked : bool | Проверена ли клетка текущим игроком. |
| Класс: UDPCommunicator | | |
| Порт для приёма сообщений | listenPort : quint16 | Порт для прослушивания сообщений. |
| Порт для отправки сообщений | sendPort : quint16 | Порт для отправки сообщений. |
| Сетевой сокет | udpSocket\_ : QUdpSocket | Сокет для работы с сетью. |
| Класс: MenuWindow | | |
| Размер поля | fieldSizeInput : QLineEdit\* | Поле ввода размера игрового поля. |
| Название игры | gameNameInput : QLineEdit\* | Поле ввода имени игры. |
| IP-адрес | ipInput : QLineEdit\* | Поле ввода IP-адреса. |
| Порт | portInput : QLineEdit\* | Поле ввода порта. |
| Имя игрока | niknameInput : QLineEdit\* | Поле ввода имени игрока. |

Таблица отношений между классами:

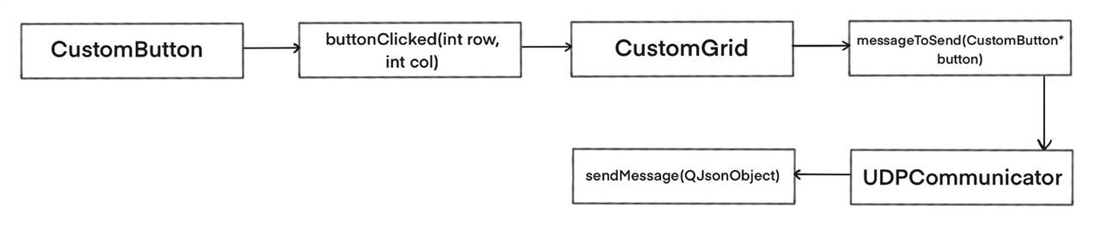
|  |  |  |
| --- | --- | --- |
| Класс с исходящей стрелкой | Класс со входящей стрелкой | Вид отношения |
| GameWindow | CustomGrid | Композиция |
| GameWindow | UDPCommunicator | Агрегация |
| CustomGrid | CustomButton | Композиция |
| UDPCommunicator | GameWindow | Ассоциация |
| MenuWindow | GameWindow | Зависимость |

2.3 Перечень библиотечных конструкторских классов, использованных в проекте

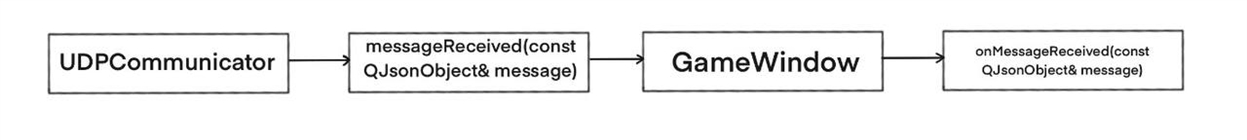
|  |  |
| --- | --- |
| **Наименование** | **Назначение** |
| |  |  | | --- | --- | | QApplication |  | | |  |  | | --- | --- | |  | Управление жизненным циклом приложения. | |
| |  | | --- | | QWidget | | |  | | --- | | Базовый класс для всех оконных виджетов. | |
| |  |  | | --- | --- | | QPushButton |  | | |  |  | | --- | --- | |  | Реализация кнопок для клеток игрового поля. | |
| |  |  | | --- | --- | | QGridLayout |  | | |  |  | | --- | --- | |  | Организация сетки виджетов на экране. | |
| |  |  | | --- | --- | | QVector |  | | |  |  | | --- | --- | |  | Хранение списков кнопок и других объектов. | |
| QSet | |  |  | | --- | --- | |  | Хранение уникальных выбранных кнопок на игровом поле. | |
| |  |  | | --- | --- | | QLineEdit |  | | |  |  | | --- | --- | |  | Поля ввода для параметров игры (например, имя игрока, IP). | |
| |  |  | | --- | --- | | QMessageBox |  | | |  |  | | --- | --- | |  | Отображение диалоговых окон для уведомлений. | |
| |  |  | | --- | --- | | QUdpSocket |  | | |  |  | | --- | --- | |  | Организация сетевого взаимодействия через UDP-протокол. | |
| |  |  | | --- | --- | | QJsonObject |  | | |  |  | | --- | --- | |  | Форматирование данных для передачи в JSON-формате. | |
| |  |  | | --- | --- | | QJsonDocument |  | | |  |  | | --- | --- | |  | Сериализация/десериализация JSON-объектов для сетевого обмена. | |
| |  |  | | --- | --- | | QHostAddress |  | | |  |  | | --- | --- | |  | Управление IP-адресами для соединений. | |
| |  |  | | --- | --- | | QString | . | | |  | | --- | | Работа со строками текста. | |
| QLabel | Отображение заголовков |
| QHBoxLayot | Организация окон приложения |
| QByteArray | Работа с бинарными данными, передача данных по сети |
| QJsonArray | Работа с массивами json |

2.4 Соединения «сигнал-слот» реализованные в проекте

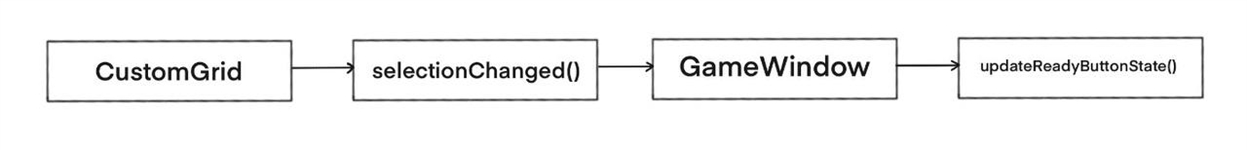
Описания конкретных действий  
1. Ход игрока (нажатие на кнопку)

  
Рисунок 3 – соединения сигналов при действии ход игрока

2. Получение сообщения от приложения соперника

  
Рисунок 4 – соединение сигналов при получении сообщения

3. Осуществление хода (изменение кнопки выбора на сетке)

  
Рисунок 5 – соединение сигналов после хода

Сигналы и слоты

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Сигнал** | **Слот** | **Назначение** |
| CustomButton | buttonClicked(int row, int col) | CustomGrid::buttonClicked(int row, int col) | Передача информации о нажатии кнопки в игровую сетку. |
| CustomGrid | messageToSend(CustomButton\* button) | UDPCommunicator::sendMessage(QJsonObject) | Отправка данных о ходе игрока. |
| CustomGrid | selectionChanged() | GameWindow::updateReadyButtonState() | Обновление состояния кнопки в зависимости от выбора. |
| UDPCommunicator | messageReceived(const QJsonObject& message) | GameWindow::onMessageReceived(const QJsonObject& message) | Обработка входящего сообщения. |
| UDPCommunicator | messageToGame(const QJsonObject& message) | GameWindow::onGameMessage(const QJsonObject& message) | Передача игровых данных между игроками. |
| GameWindow | readyClicked() | GameWindow::onReadyClicked() | Обработка нажатия на кнопку "Начать игру". |
| GameWindow | resetClicked() | GameWindow::onResetClicked() | Обработка нажатия на кнопку "Сброс". |

3. Подсистема «Интерфейс»

3.1 Графическое представление интерфейсных окон:

1. Главное окно меню:

- Радио-кнопки  
 -для локальной игры  
 -поле для ввода размера поля

-для создания игры  
 -поля для ввода размера поля, названия игры, IP-адреса, порта, имени игрока.  
 -для подключения к игре  
 -поля для IP-адреса, порта, имени игрока.

- Кнопка для запуска игры.

- Кнопка для выхода из программы

2. Игровое окно:

- Два поля (игрока и соперника).  
 Каждое состоит из:  
 -подписей где чье поле  
 -двух квадратных полей клеток заданной ширины

-кнопки готовности к игре

-кнопки сброса выбранных клеток  
3. Окна уведомлений

-Диалоговые окна состоящие из сообщения и кнопки подтверждения

4. Подсистема «Модель»

4.1 Перечень событий, изменяющих состояние модели

|  |  |
| --- | --- |
| **Событие** | **Последовательность действий** |
| **Внешние (пользовательские)** | |
| Начальное состояние | |  | | --- | | - Создание игровых полей для игроков. |  |  |  | | --- | --- | |  | - Генерация исходного состояния поля. |  |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | |  | | --- | | - Игрок размещает числа на своём поле. |  |  |  | | --- | --- | |  | - Поле соперника отображается как закрытое. | | |
| Пуск игры | |  | | --- | | - Игрок нажимает кнопку "Готов". |  |  |  | | --- | --- | |  | - Отправляется сигнал готовности сопернику через UDPCommunicator. - Определяется первый игрок | |  |  | |
| Ход игрока | |  | | --- | | - Игрок выбирает клетку на поле противника (нажатие на кнопку). |  |  |  | | --- | --- | |  | - Состояние клетки обновляется (попадание/промах). |  |  |  | | --- | --- | |  | - Информация отправляется сопернику через UDPCommunicator. | |
| Конец игры | |  | | --- | | - По окончании ходов или открытии всех выбранных клеток игра завершается. |  |  |  | | --- | --- | |  | - Подводятся итоги, отображается победитель. |  |  |  | | --- | --- | |  | - Отправка сетевого сообщения о завершении игры. | |
| **Внутренние (программные)** | |
| Обработка хода | - Получение сетевого сообщения через UDPCommunicator.   |  | | --- | | - Обновление состояния клетки на поле игрока. |  |  |  | | --- | --- | |  | - Проверка завершения игры. | |
| Обновление состояния | - Обновление состояния |

4.2 Заголовочные файлы классов модели:

|  |  |
| --- | --- |
| Название файла | Описание |
| gamewindow.h | координирует игровой процесс. |
| menuwindow.h | предоставляет интерфейс для ввода параметров игры и запуска процесса. |
| customgrid.h | реализует игровое поле и взаимодействие между клетками. |
| udpcommunicator.h | отвечает за передачу данных между игроками. |
| custombutton.h | содержит атрибуты клетки и методы управления |

**Основное окно** – файл gamewindow.h, класс GameWindow:  
Поля:  
- CustomGrid \*playerGrid;   
- CustomGrid \*opponentGrid;   
- UDPCommunicator \*communicator;   
- QPushButton \*readyButton;   
- QString nikname;   
- QString gameName;   
- int turn;  
- bool isGameCreator;  
- qint16 receivePort;

Прототипы методов:  
- GameWindow(const QString &title, const QString &nikname, int fieldSize, quint16 sendPort, bool isGameCreator, QWidget \*parent = nullptr, const QString &ip = "");

- CustomGrid\* getPlayerGrid();

- CustomGrid\* getOpponentGrid();

- UDPCommunicator\* getCommunicator();

- void setTurn(int t\_turn);

- getTurn();

- QString getNikname();

- void setNikname(QString t\_nikname);

- QString getGameName();

- void setGameName(QString t\_gameName);

- void updateGrids(int newSize);

- void onSendMessage();

- void onReadyClicked();

- void onResetClicked();

- updateReadyButtonState();

- void ButtonChecked(CustomButton\* button);

- void onMessageReceived(const QJsonObject& message);   
  
  
**Окно параметров** – файл menuwindow.h, класс MenuWindow:  
Поля:  
- QLineEdit \*fieldSizeInput;   
- QLineEdit \*gameNameInput;   
- QLineEdit \*ipInput;   
- QLineEdit \*portInput;   
- QLineEdit \*niknameInput;

Прототипы методов:   
- MenuWindow(QWidget \*parent = nullptr);

**Игровое поле** – файл customgrid.h, класс CustomGrid:  
Поля:  
- int rows, cols;   
- QVector<CustomButton \*> buttons;   
- QSet<CustomButton \*> selectedButtons;  
- QGridLayout \*layout;   
- bool enemyGrid;   
- bool isGameCreator;   
- int &turn;  
- int rightButtons;  
- int rightFoundedButtons;  
- int summ;

Прототипы методов:   
- CustomGrid(bool isGameCreator, bool enemyGrid, int rows, int cols, int &turn, QWidget \*parent = nullptr);   
- ~CustomGrid();  
- void createButtons();  
- void buttonClicked(int row, int col);  
- void addToRightButtons(int num);  
- void addToSumm(int num);  
- int getSumm();  
- int getRightButtons();  
- int getRightFoundedButtons();  
- void updateGrid(int newRows, int newCols);   
- CustomButton\* getButtonAt(int row, int col);   
- void updateEnemyGrid(const QVector<QVector<int>>& fieldData);   
- void toggleButtonSelection(CustomButton \*button);

**Сетевой коммуникатор** – файл udpcommunicator.h, класс UDPCommunicator:

Поля:   
- GameWindow\* gameWindow;  
- quint16 listenPort;   
- quint16 sendPort;   
- QUdpSocket udpSocket\_;   
- QHostAddress remoteAddress\_;

Прототипы методов:   
- UDPCommunicator(quint16 listenPort, quint16 sendPort, const QString &ip, QObject \*parent = nullptr);   
- void setGameWindow(GameWindow\* t\_gameWindow);   
- void sendMessage(const QJsonObject& message);   
- void startReceiving();  
- QJsonObject createFieldReadyMessage(CustomGrid\* grid);   
- QJsonObject createFieldMessage(int turn, CustomButton\* button);   
- QJsonObject createWinMessage(const QString& name);   
- QJsonObject createGetGeneralFieldDataMessage();  
- QJsonObject createGeneralFieldDataMessage(const QString& name, int fieldSize);   
- QJsonObject createGetSummDataMessage();  
- QJsonObject createSummDataMessage(int summ, const QString& name);   
- void parseFieldReadyMessage(const QJsonObject& message);   
- void parseFieldMessage(const QJsonObject& message);   
- void parseWinMessage(const QJsonObject& message);   
- void parseGeneralFieldDataMessage(const QJsonObject& message);   
- void parseGetSummMessage(const QJsonObject& message);   
- void processReceivedMessage(); (слот)

**Клетки –** файл custombutton.h, класс CustomButton  
Поля:   
- int value;  
- int row;  
- int col;   
- bool alreadyChecked;   
- bool checkedByEnemy;

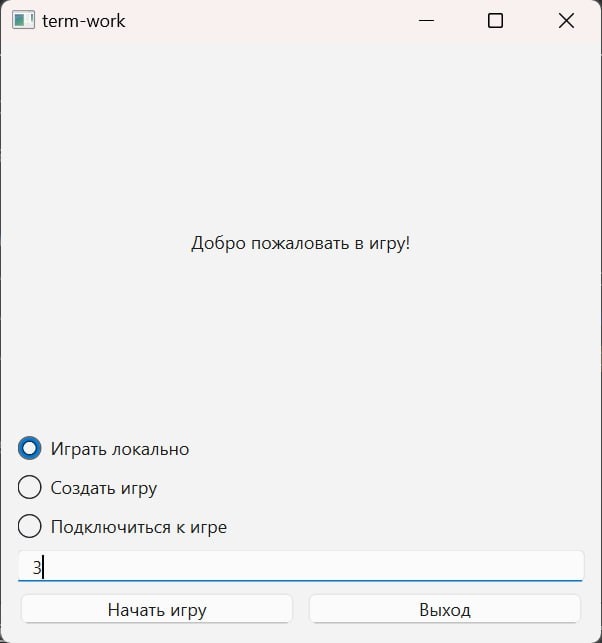
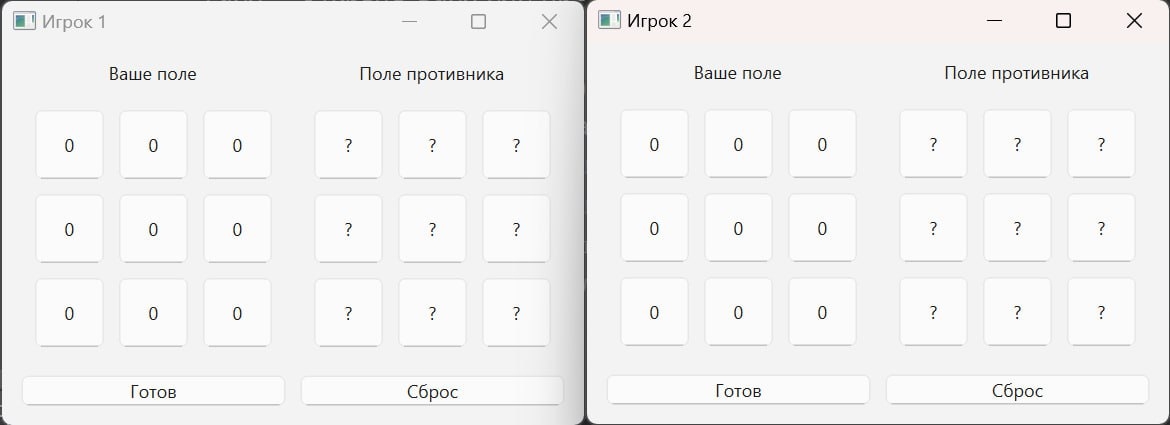
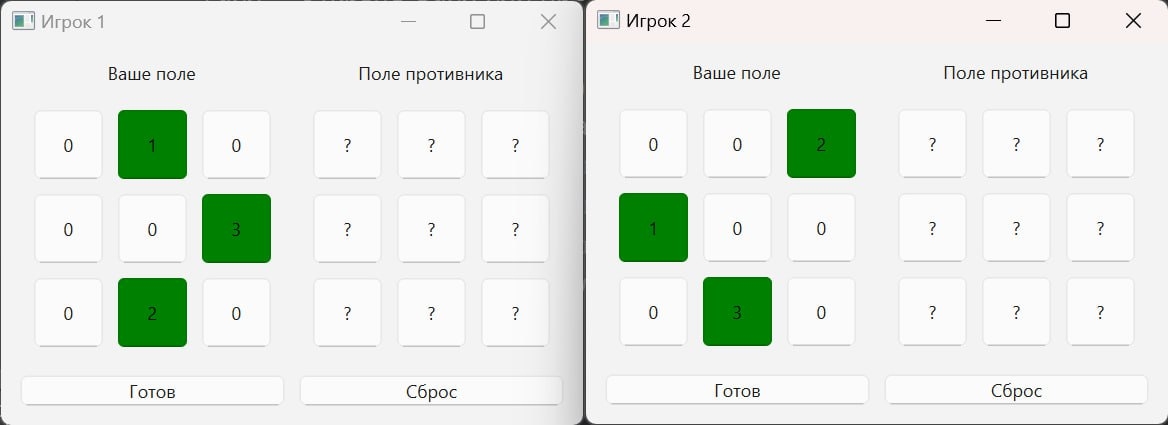
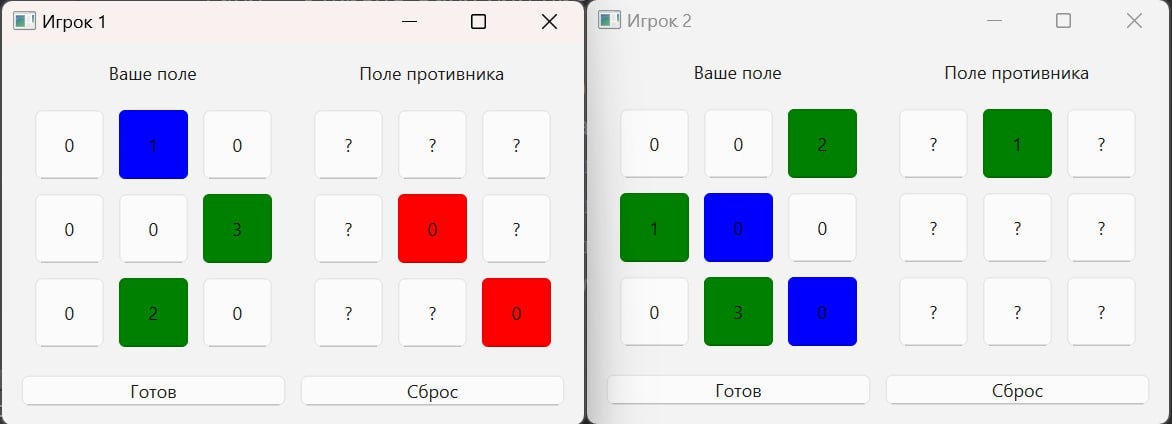
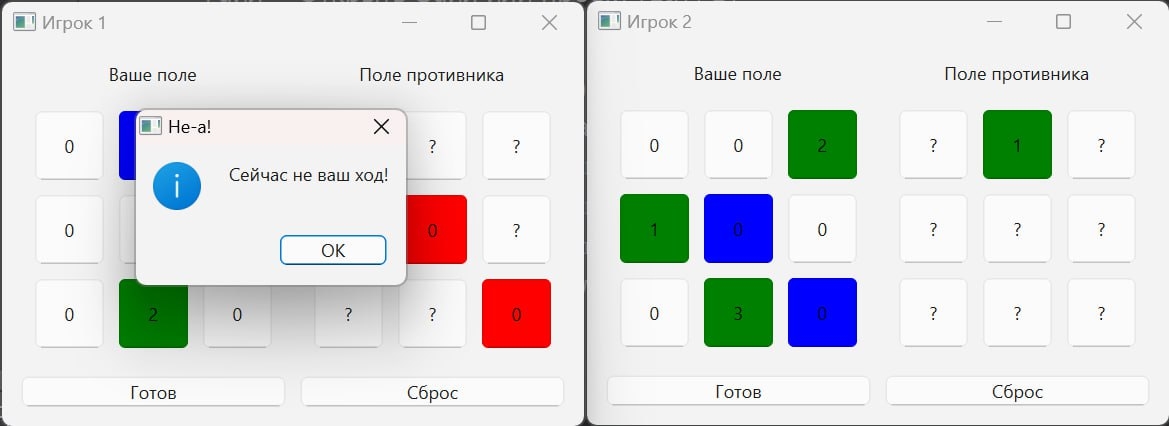
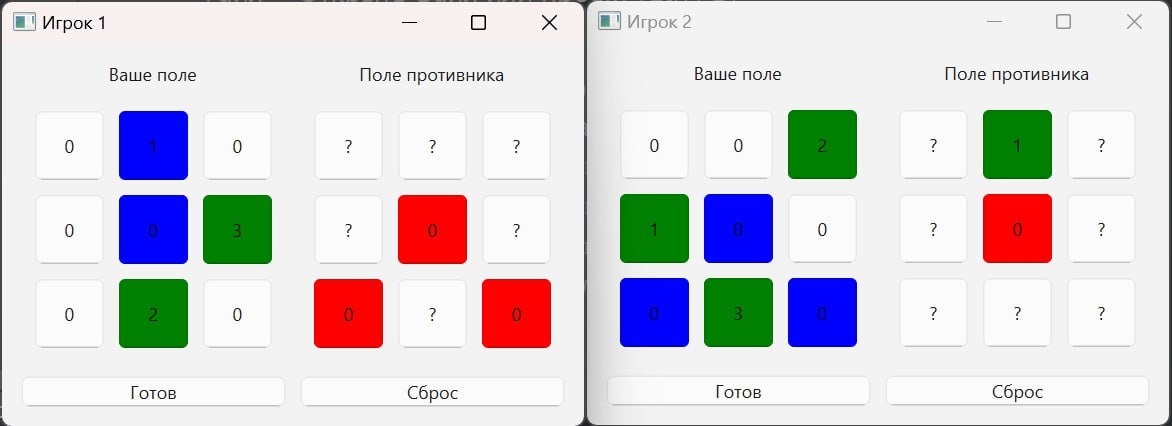
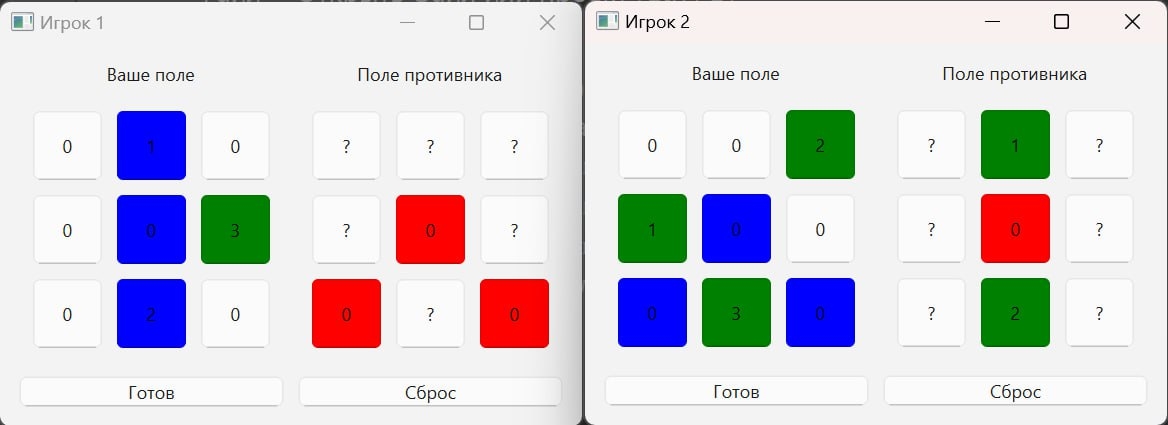
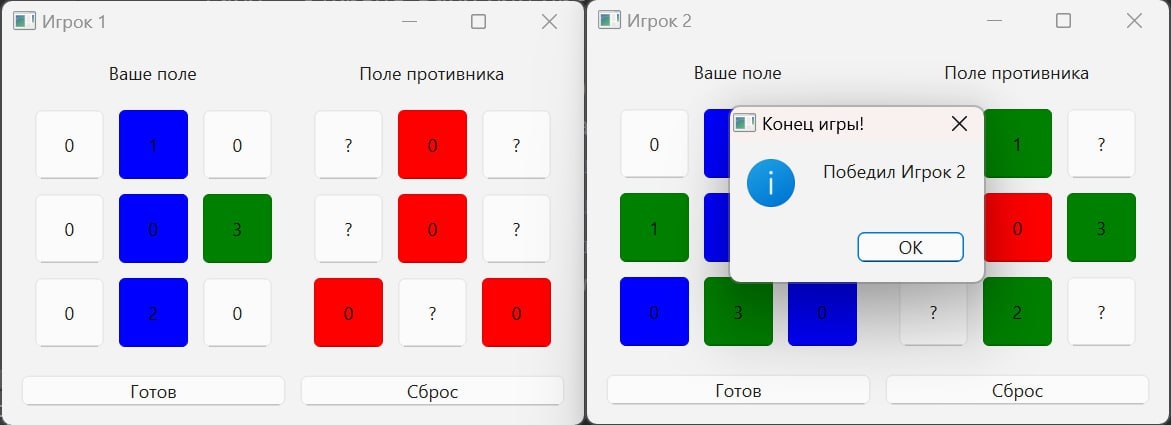
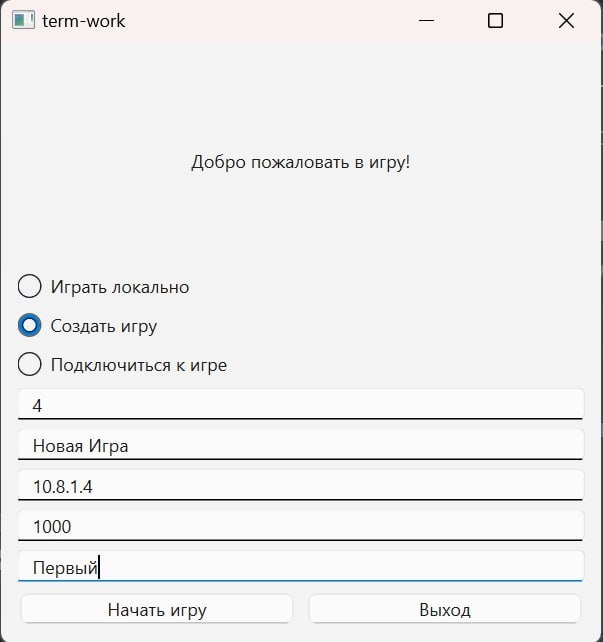
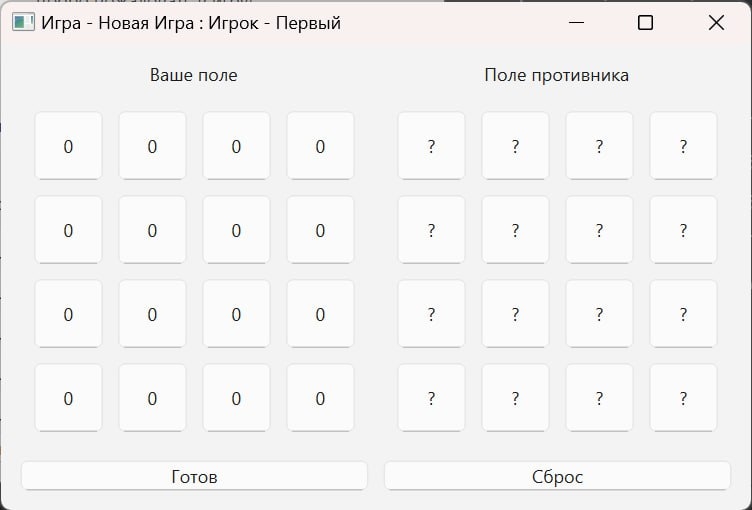
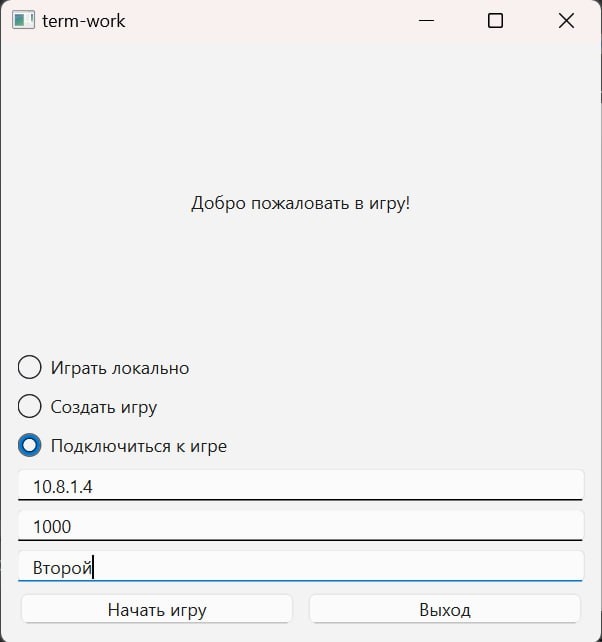
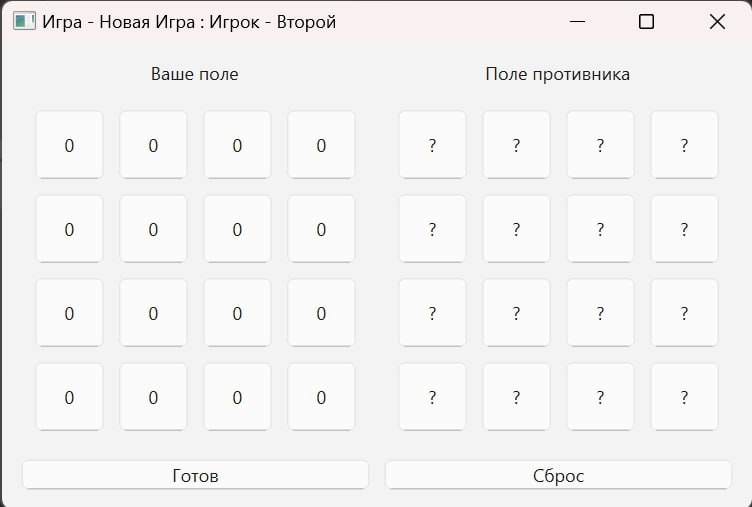
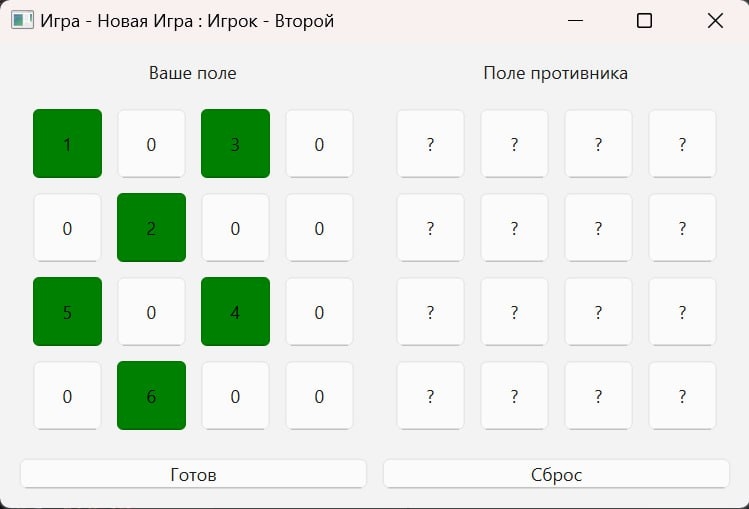
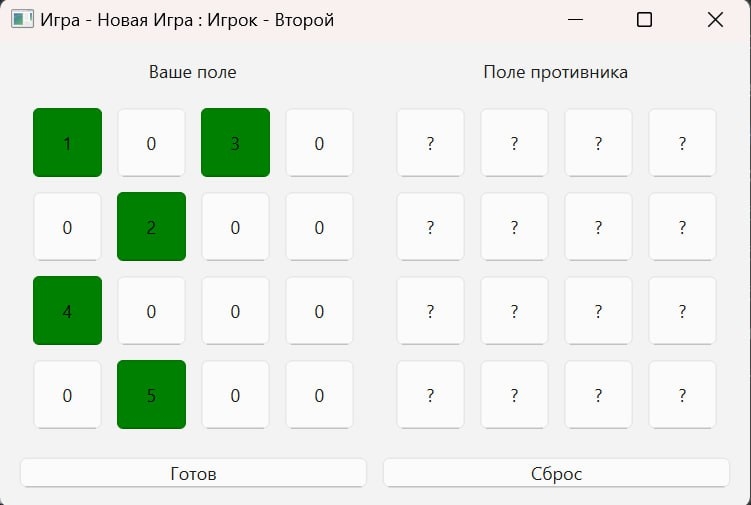
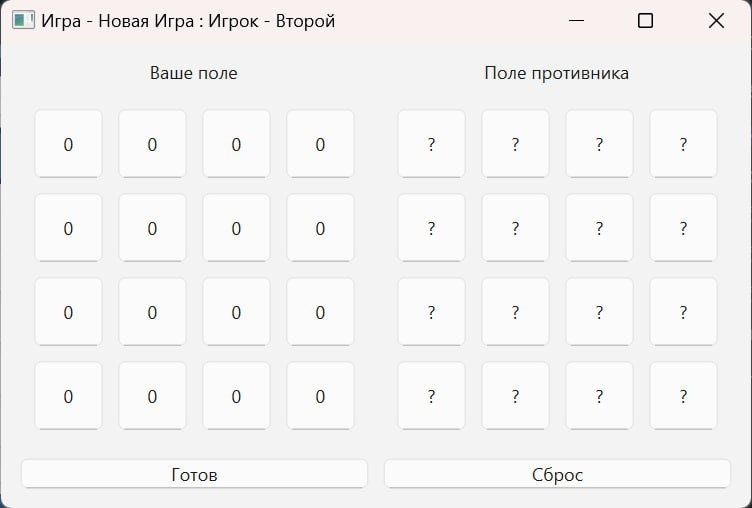
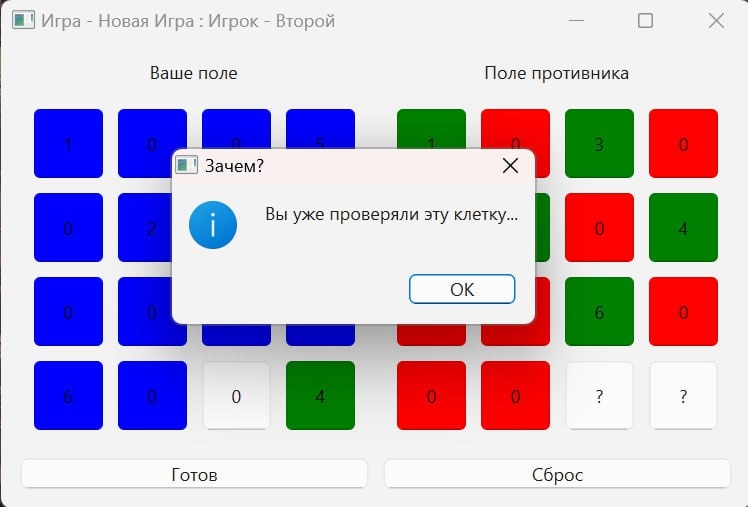
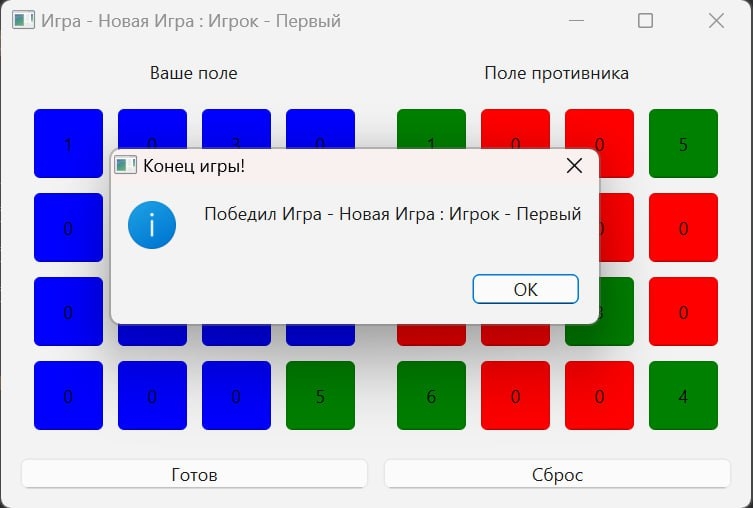
Прототипы методов:   
- CustomButton(int value, int row, int col, QWidget \*parent = nullptr);   
- ~CustomButton();  
- int getRow() const;  
- int getCol() const;   
- int getValue() const;   
- void setValue;  
- void setCheckedByEnemy(bool checked);   
- void setAlreadyChecked(bool checked);   
- void setButtonStyle();  
- bool isAlreadyChecked() const;

5. Контрольный пример для реализованного приложения

5.1 Контрольный пример  
Цель контрольного примера – проверить правильность работы программы.  
  
Контрольный пример:

1. Запустим приложение
2. Выберем локальный вариант игры с параметрами: размер поля 3
3. Начнем игру
4. Выберем по три клетки для каждого игрока  
   если рассматривать как матрицу: игрок 1 (0, 1), (2, 1), (1, 2) и   
   игрок 2 (1, 0), (0, 2), (2, 1)
5. Сделаем несколько ходов для демонстрации – попадания отмечаются зеленым цветом на поле противника, промахи – синим
6. Попробуем сходить два раза подряд
7. Продемонстрируем поведение попадания
8. Победа от всех выбранных клеток
9. Выберем создание сетевого варианта игры с параметрами: размер поля - 4, Название – Новая Игра, IP 10.8.1.4, порт – 1000, имя игрока Первый
10. Начнем игру
11. Теперь выберем подключится к игре
12. Введем IP - 10.8.1.4, порт – 1000, имя игрока Второй
13. Начнем игру
14. Продемонстрируем изменение положения чисел в выбранных клетках
15. Выберем клетки (0, 0), (1, 1), (0, 2), (2, 2), (2, 0), (3, 1)
16. Уберем клетку (2, 2) со значением 4
17. Клетки (2, 0), (3, 1) должны поменять свои значения на 4 и 5 соответственно
18. Сбросим выбранное
19. После нескольких ходов попробуем выбрать уже открытую клетку
20. Победа в сетевой игре

Скриншоты работы программы на основе контрольного примера

1. Запустим приложение  
   
2. Запустим игру в соответствии с контрольным примером
3. Выберем клетки с соответствии с контрольным примером
4. Продемонстрируем изменение выбранных клеток
5. Попробуем сходить два раза подряд
6. Продемонстрируем поведение клетки при попадании  
   состояние до хода 
7. Состояние после хода  
   
8. Победа  
   
9. Выберем создание сетевого варианта игры в соответствии с контрольным примером  
   
10. Запустим
11. Выберем подключение к игре  
    
12. Запустим  
    
13. Выберем клетки в соответствии с контрольным примером  
    
14. Уберем клетку с цифрой 4  
    
15. Сбросим  
    
16. Проверим клетку дважды  
    
17. Победа в сетевой игре  
    

Результаты работы программы совпадают с ожидаемыми, программа работает корректно.

5.2 Инструкция для пользователя по работе с программой:

1. Запустите приложение.
2. Выберите тип игры
   1. Локальная
      1. Введите размер поля.
   2. Сетевая (создание)
      1. Введите размер поля, название игры, IP-адрес, порт, имя игрока.
   3. Сетевая (подключение)
      1. Введите IP-адрес, порт, имя игрока.
3. Расставьте числа на поле.
4. Нажмите кнопку готов и ждите начала игры.
5. Делайте ходы, открывая клетки соперника.
6. По завершении игры посмотрите результаты.

6. Выводы

В результате курсовой работы разработана и протестирована сетевая игра с использованием объектно-ориентированного подхода. Программа отвечает заданию: поддерживается сетевое взаимодействие, возможно работа на одном устройстве, соблюдается обработка правил игры, взаимодействие происходит через GUI. Программа демонстрирует корректную работу основных компонентов и взаимодействие игроков через сеть.   
  
Язык реализации С++. Объектная библиотека для выбора конструкторских классов — Qt. Среда реализации - Qt Creator.