Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатики и систем управления»

КАФЕДРА «Защиты информации»

РАСЧЕТНО−ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка программы "Морской бой"

Студент *ИУ10-21* _____ *Ф.А. Козиев*
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсового проекта _____ *А.В. Астрахов*
(Подпись, дата) (И.О. Фамилия)

Консультант _____ *И.Р. Матакаев*
(Подпись, дата) (И.О. Фамилия)

2021 г.

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой _____
(Индекс)
_____ _____
(И.О. Фамилия)
« ____ » _____ 20 ____ г.

З А Д А Н И Е
на выполнение курсовой работы

по дисциплине *Информатика*

Студент группы *ИУ10-21*

Козиев Фёдор Александрович
(Фамилия, имя, отчество)

Тема курсовой работы *Разработка программы "Морской бой"*

Направленность КР (учебная, исследовательская, практическая, производственная, др.)
учебная
Источник тематики (кафедра, предприятие, НИР) *кафедра*

График выполнения КР: 25% к *4* нед., 50% к *9* нед., 75% к *12* нед., 100% к *15* нед.

Задание: *программа разрабатывается на языке программирования Си с использованием сетевых сокетов и должна реализовать игру "Морской бой" для двух пользователей, располагающихся в локальной сети.*

Оформление курсовой работы:

Расчетно-пояснительная записка на *60* листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)
текст программы на языке программирования Си

Дата выдачи задания « *19* » *февраля* 20*21* г.

Руководитель курсового проекта _____ *А.В. Астрахов*
(Подпись, дата) (И.О. Фамилия)

Студент _____ *Ф.А. Козиев*
(Подпись, дата) (И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Морской бой – интересная настольная игра для двух участников, в которой игроки по очереди называют координаты на неизвестной им карте соперника. На языке программирования СИ можно написать приложение для того, чтобы два пользователя могли играть в эту игру, находясь в одной локальной сети.

Целью работы является разработка клиент-серверного приложения для игры в «Морской бой» (с графическим интерфейсом написанном на Qt Framework).

Задачи, которые необходимо решить для написания программы:

1. Ознакомление с подробными правилами игры «Морской бой».
2. Написание основных функций для работы логической части программы.
3. Изучение материалов о UDP протоколе.
4. Написание полноценного клиент – серверного приложения для игры в морской для двух пользователей, находящихся в локальной сети.
5. Изучение документации, продумывание связи логики игры, логики клиент – серверного приложения и графического интерфейса.
6. Написание графического интерфейса при помощи Qt Framework.

Результатом является клиент-серверное приложение с графическим интерфейсом для игры двух пользователей в «Морской бой», находящихся в локальной сети.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Прежде чем начать писать логику игры «Морской бой» необходимо изучить правила этой игры. В курсовой работе будут использованы правила классического морского боя.

Основные правила игры:

1. У каждого игрока 2 поля 10x10. На одном находятся его корабли, а на другом результаты его выстрелов.
2. Вертикально происходит нумерация сверху вниз, горизонтально помечается буквами слева направо.
3. Прежде чем начать игру оба пользователя должны расставить на поле свои корабли по следующим правилам и в следующем количестве:
    a. На поле должно быть расставлено 4 корабля занимающие одну клетку, 3 корабля занимающие две клетки, 2 корабля занимающие три клетки и 1 корабль, занимающий 4 клетки.
    b. Корабли могут ставиться только горизонтально или вертикально (без изгибов и не по диагонали).
    c. Вокруг корабля на расстоянии одной клетки по всем направлениям нельзя ставить другой корабль.
4. После расстановки кораблей игроки по очереди выбирают конкретную клетку, по которой производят «выстрел». Говорят координаты этой клетки другому игроку, и другой игрок должен сказать результат «выстрела»: «промах», если выбранная клетка не принадлежит ни одному из его кораблей, «попал», если выбранная клетка принадлежит одному из его кораблей и при этом у этого корабля еще остались клетки, по которым еще не был произведен «выстрел», «убил», если выбранная клетка принадлежит одному из кораблей и у этого корабля больше нет клеток, по которым не был произведен «выстрел».

5. Если игрок «попал» или «убил», то он выбирают клетку снова, если результатом выстрела оказался «промах», то ход переходит к другому игроку.
6. Выигрывает тот игрок, который первый уничтожит все корабли соперника.

При написании клиент-серверной части курсовой работы необходимо написать программу для клиента и для сервера:

Клиент – компонент программы, посылающий и принимающий сообщения от сервера. Отправка и получение сообщений происходит по определенному протоколу. Клиентов, как правило, больше, чем серверов.

Сервер – является «центральной» частью клиент – серверного приложения, часто хранит данные и к нему обращаются множество клиентов. Можно сказать, что сервер может существовать без клиентов, а вот клиенты без сервера не могут.

Для «общения» между собой сервер и клиент используют определенный протокол, т. е. набор соглашений, который определяет обмен данными между сервером и клиентом. Протоколы задают способы передачи сообщений и обработки ошибок в сети, а также позволяют разрабатывать стандарты, не привязанные к конкретной аппаратной платформе.

В курсовой работе будет использоваться протокол UDP, который очень распространён в современном мире и используется для передачи датаграмм, т. е. независимый пакет данных, несущий информацию, достаточную для доставки сообщения от отправителя до получателя. Данный протокол не требует установки соединения и не гарантирует целостность и правильность доставленного пакета, за счет этого дает возможность очень быстро передавать пакеты.

Для переданных данных по сети используются сокеты. Сокет – это средство операционной системы, необходимое для того, чтобы обеспечивать отправку и получение данных вне зависимости от протокола.

В курсовой работе создадим две UML – диаграммы. UML-диаграмма последовательностей, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие действующих лиц информационной системы. И UML-диаграмма вариантов использования, на

которой отображается отношение между всеми действующими лицами, эту диаграмму используют для демонстрации работы программы пользователю, который не разбирается в данной теме.

В данную курсовую работу будет интегрирован графический интерфейс, написанный при помощи Qt Framework.

Qt Framework - фреймворк для разработки кроссплатформенного программного обеспечения на языке программирования C++.

После того, как описаны основные теоретические понятия, можно перейти к написанию логики игры, клиент-серверной части и графической оболочки.

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

Прежде чем начать писать клиент-серверную часть и графическую оболочку стоит написать логику игры, рассмотрев следующие моменты:

1. Способ представления поля.
2. Способ отображения поля.
3. Алгоритмы расстановки кораблей на поле.
4. Алгоритм проверки корректности хода.
5. Алгоритм обработки хода.
6. Алгоритм обработки результата хода («убийства» корабля).
7. Алгоритм определения есть ли живые корабли.
8. Алгоритмы обработки убийства корабля

Поле будет представлено в виде матрицы 10x10, которая содержит числа: 0 – свободное поле, 1 – поле, соседнее с кораблем, которое нельзя занимать, 2 – поле принадлежащее кораблю, 3 – поле, по которому был произведен выстрел, результатом которого стал промах, 4 – поле, по которому был произведен выстрел, результатом которого стало попадание, 5 – поле, по которому был произведен выстрел, результатом которого стало «убийство» корабля.

Отображение поля будет происходить путем закрашивания клетки тем, или иным цветом. Если клетка свободна, она закрашивается зеленым, если клетка соседняя с кораблем, то она закрашивается желтым, если клетка принадлежит какому-то кораблю, то она закрашивается синим, если по клетке был произведен выстрел, результатом которого стал промах, то она закрашивается красным, если по клетке был произведен выстрел, результатом которого стало попадание по кораблю, то клетка закрашивается фиолетовым, если же по клетке был произведен выстрел, и она имеет статут «убит», то она закрашивается бордовым.

Алгоритм расстановки кораблей на поле заключается в том, что необходимо ввести координаты начала корабля и координаты конца корабля, после чего текущее размещение корабля проверяется на корректность, если размещение корректно, то выполняется установка корабля с учетом расстановки соседних полей, которые нельзя занимать.

Алгоритм проверки корректности хода после разработки графического интерфейса, исключающего возможность ввода координаты, выходящей за границы поля, облегчился до того, что необходимо проверять только то, какое состояние у выбранной клетки, если оно не равно 0, т. е. если клетка не пустая, то ход считается некорректным.

Для обработки хода необходимо сперва проверить состояние выбранной клетки, если оно равно 0 или 1, т. е. если клетка пустая или соседняя с кораблем, то результатом хода является промах, если состояние выбранной клетки равно 2, т. е. если клетка принадлежит какому то кораблю, то запускается проверка проверки на то, «убит» ли корабль (алгоритм будет описан ниже). Если корабль «убит», то все поля убитого корабля помечаются как «убитые» и на поле, где отмечаются выстрелы игрока, вокруг полей, принадлежащих уничтоженному кораблю, заполняются соседние поля для того, чтобы игрок не «стрелял» по тем полям, где точно не может быть корабля. Если же судно не было уничтожено, то текущее поле принимает значение «попал», то есть результатом выстрела является попадание.

Алгоритм определения есть ли живые корабли - простая проверка: в цикле идем по всем ячейкам поля, если находим поле, состояние которого равно 2, т. е. если поле принадлежит какому-то кораблю, то становится очевидно, что игрок, у которого ведется проверка не проиграл.

Алгоритмы обработки убийства корабля основываются на том, что от выбранной позиции рассматриваются поля по 4 направлениям, и если в одном из направлений есть поле, имеющее статус «попал», то продолжается «движение» по этому направлению, иначе начинается движение по другому направлению.

После написания основной логики, приступим к написанию клиент-серверная часть программы. Клиент-серверная составляющая курсовой работы будет представлена двумя программами – клиентом и сервером. Начнем разработку функций клиента.

Необходимые алгоритмы:

1. Инициализации всех данных необходимых для отправки данных на сервер

2. Алгоритм для отправки на сервер имени пользователя и получения имени другого пользователя.
3. Алгоритм для получения текущего состояния клиента
4. Алгоритм необходимая для отправки хода и получения подтверждения того, что ход пришел от сервера
5. Алгоритм для получения хода от сервера
6. Алгоритм для отправки результата обработки хода
7. Алгоритм для получения результата обработки хода

Вся программа клиента выполняется, опираясь на то, какое сейчас состояние: делает ход, ожидает ход другого игрока, победитель, проигравший.

Если сейчас пользователь делает ход, то введенный ход отправляется на сервер, принимается подтверждение корректности хода, если ход некорректный, то он будет отправляться на сервер до тех пор, пока не придет подтверждение о том, что ход корректный. Затем пользователь ожидает ответа от сервера с результатом хода. Если результат хода – промах, то состояние клиента меняется на ожидание хода.

Если сейчас пользователь ожидает ход, то сначала производится запрос хода у сервера, когда ход получен происходит его обработка и отправка на сервер, если результат обработки хода – промах, то состояние клиента меняется на выбор хода.

Теперь рассмотрим принцип работы сервера. Функции необходимые для работы сервера похожи на те, которые необходимы для работоспособности клиента.

Сперва происходит инициализация всех данных для отправки и получения данных, затем происходит регистрация пользователей: получение имени пользователя первого игрока, отправка ему того, что он будет ходить, когда будет добавлен второй пользователь. Затем происходит получение имени пользователя второго игрока, отправка ему того, что он будет ожидать ход, отправка ему имени пользователя первого игрока и отправка первому игроку имя пользователя второго игрока. После того, как оба пользователя зарегистрированы начинается игра. Сперва получаем ход от первого игрока, отправляем ему результат проверки

корректности хода, если ход некорректный, то запрашиваем ход до тех пор, пока не придет корректный ход. После получения корректного хода мы отправляем ход второму игроку, получаем от него результат хода и отправляем его первому игроку, если результат хода «промах», то текущий игрок меняется на второго игрока и выполняются такие же действия для второго игрока.

В данной курсовой работе будет интегрирован графический интерфейс, написанный при помощи Qt Framework.

Каждая клетка является классом Cell переопределенным от QPushButton. Полями класса является две координаты, для отправки координат хода мы переопределяем метод нажатия на кнопку. Если кнопка нажата, то отправляется сигнал нажатия кнопки с координатами этой кнопки.

Каждое поле является классом Field, наследованным от QWidget в которое входит 100 объектов класса Cell. В конструкторе этого класса создаем каждую клетку и добавляем ее в QGridLayout в соответствии с текущими координатами. И связываем сигнал каждой клетки с слотом получения хода, этот слот вызывает сигнал с координатами клетки, вызвавшей этот слот.

Сама логика игры написана на языке программирования СИ, но для корректной и удобной работы графического интерфейса необходимо написать «класс - обертку». Класс Game внутри своих методов вызывает необходимые функции, которые добавлены в заголовочный файл с помощью спецификатора extern “C” и в файл исходных текстов.

2 объекта класса Field являются полями основного класса MainWindow. Поле с кораблями и поле с выстрелами соответственно. Так же полями этого класса является объект класса Game и объект Qtimer, который необходим для того, чтобы во время ожидания хода или имени пользователя другого игрока приложение не «зависало». QTimer вызывает слот _on_timer_tick каждые 100мс, который в зависимости от того, что сейчас ожидает программа вызывает соответствующий метод класса Game.

В классе MainWindow так же реализована расстановка кораблей по следующему алгоритму: если не все корабли расставлены, то проверяется в слоте, вызванном классом Field введена ли начальная координата, если нет, то

полученная координата записывается в как начальная координата, если начальная координата уже введена, то полученная координата записывается как конечная координата. После того, как обе координаты введены запускается проверка корректности положения корабля, и в зависимости от результата выполняется установка корабля или сброс координат. Также при расстановке учитывается ограничение количества кораблей с каждой длинной. Начальную координату при расстановке кораблей можно сбросить нажатием клавиши Esc. Когда все корабли расставлены необходимо отправить имя пользователя (с помощью специальной кнопки или клавиши Enter) и дождаться имя пользователя второго игрока, после его получения начинается игра по алгоритмам, описанным выше.

ЗАКЛЮЧЕНИЕ

В результате написания курсовой работы было получено клиент-серверное приложение с графическим интерфейсом, позволяющее двум пользователям, находящихся в локальной сети, играть в «Морской бой».

В ходе написания программы были закреплены навыки программирования на языке СИ, углублены знания Qt Framework. Также был изучен принцип работы клиент – серверного приложения с использованием протокола UDP. Помимо всего прочего было освоено создание UML диаграмм.

СПИСКА ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ ИНФОРМАЦИИ

1. https://ru.stackoverflow.com
2. https://coderoad.ru
3. https://ru.wikipedia.org/wiki/Морской_бой_(игра)
4. http://book.itep.ru/4/44/udp_442.htm

ПРИЛОЖЕНИЕ А. UML – ДИАГРАММЫ

Диаграмма последовательностей UML:

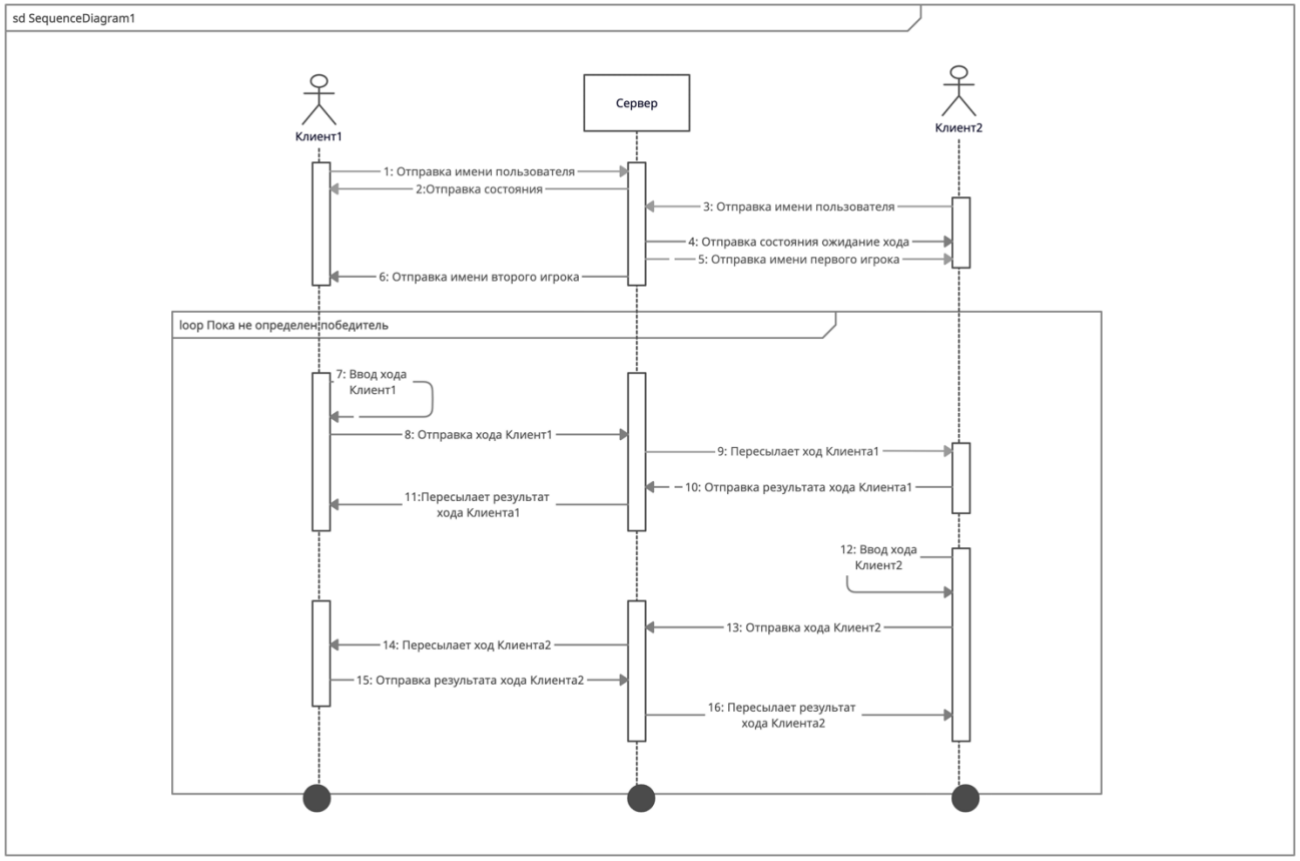
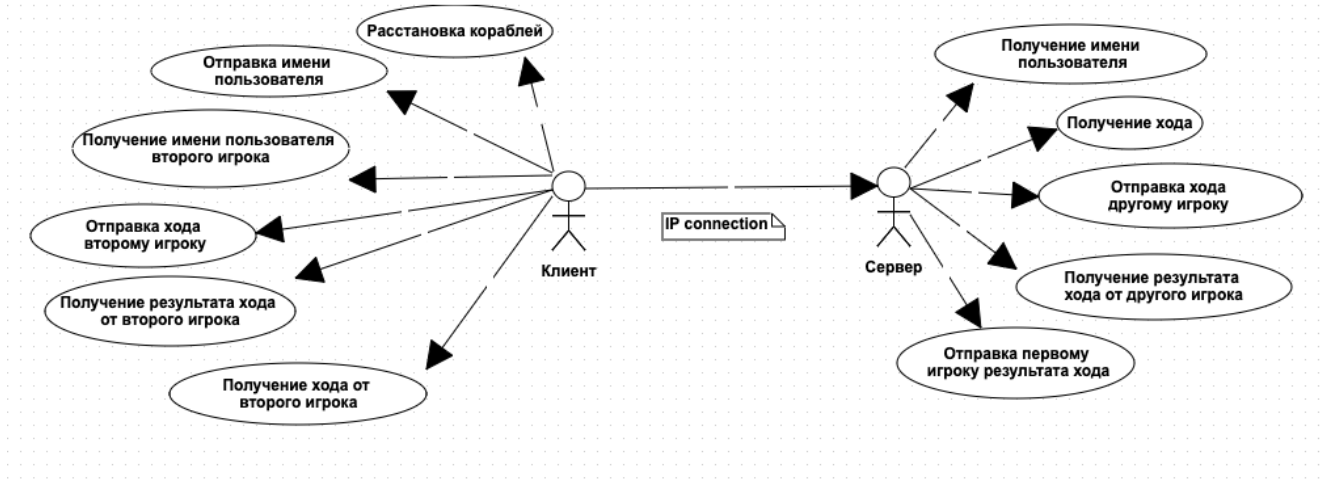


Диаграмма вариантов использования UML:



ПРИЛОЖЕНИЕ Б. ИСХОДНЫЕ ТЕКСТЫ ПРОГРАММ.

Клиент:

Файл «_game.h»:

```
#ifndef _GAME_H
#define _GAME_H

#ifdef __cplusplus
extern "C"
{
#endif

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>

#define PORT     8080
#define MAXLINE 1024

enum Condition
{
    ERROR = -1,
    Empty = 0,
    Neighbor = 1,
    Engaged = 2,
    Miss = 3,
    Hit = 4,
    Killed = 5
};

enum ProcessingMode
{
    Horizontal = 0,
    Vertical = 1,
    Circular_right = 2,
    Circular_left = 3,
    Circular_top = 4,
    Circular_down = 5
};

enum State
{
    Unknown = 0,
    Making_move = 1,
    Waiting_move = 2,
    Winner = 3,
    Loser = 4
};

struct Move
```

```
    {
        int line;
        int column;
    };

    typedef struct Move Move;

    struct Answer
    {
        int result_of_shoot;
        int is_any_live_ship;
    };

    typedef struct Answer Answer;

    struct NetworkContext
    {
        int sockfd;
        struct sockaddr_in servaddr;
    };

    typedef struct NetworkContext NetworkContext;

    unsigned short **_create_matrix();

    NetworkContext *_network_init(uint16_t port);

    bool _is_any_live_ship(unsigned short **matrix);

    void _change_around_killed(unsigned short **matrix, int line, int
column,
                               int processing_mode);

    bool _is_horizontal(unsigned short **matrix, int line, int column);

    void _change_after_kill(unsigned short **matrix, int line, int
column);

    bool _is_killed(unsigned short **matrix, int line, int column);

    int _result_of_shot (unsigned short **matrix, int line, int column);

    void _set_ship(unsigned short **matrix, int str1, int str2, int
column1, int column2);

    bool _is_it_possible_to_put_the_ship(unsigned short **matrix, int
str1, int str2,
                                         int column1, int column2);

    bool _is_valid_move(unsigned short **matrix, int line, int column);

    void send_move(NetworkContext *ctx, Move move);

    int receive_confirm(NetworkContext *ctx);

    Answer receive_answer(NetworkContext *ctx);

    Move _receive_move(NetworkContext *ctx);

    void send_answer(NetworkContext *ctx, Answer answer);
```

```
void _send_username(NetworkContext *ctx, char username[MAXLINE]);

int receive_state(NetworkContext *ctx);

void _receive_username(NetworkContext *ctx, char username[MAXLINE]);

#ifdef __cplusplus
}
#endif

#endif // _GAME_H
```

Файл «cell.h»:

```
#ifndef CELL_H
#define CELL_H

#include <QPushButton>

class Cell:public QPushButton
{
    Q_OBJECT

private:
    int _line;
    int _column;

public:
    explicit Cell(QObject *parent = nullptr);
    void setCoordinates(int line, int column);

signals:
    void cell_clicked(int line, int column);

protected:
    void mousePressEvent(QMouseEvent *event);

};

#endif // CELL_H
```

Файл «field.h»:

```
#ifndef FIELD_H
#define FIELD_H

#include <QWidget>
#include <QGridLayout>
#include <QLabel>
#include <cell.h>

#define FIELD_LENGTH 10
#define FIELD_WIDTH 10

class Field : public QWidget
{
    Q_OBJECT
```

```
private:
    Cell *_cells[FIELD_LENGTH][FIELD_WIDTH];

private slots:
    void get_coordinates(int line, int column);

public:
    explicit Field(QWidget *parent = nullptr);
    void update_matrix(unsigned short **matrix);

signals:
    void send_move(int line, int column);
};

#endif // FIELD_H
```

Файл «game.h»:

```
#ifndef GAME_H
#define GAME_H
#include <QWidget>
#include<_game.h>

class Game : public QWidget
{
    Q_OBJECT

private:

    unsigned short **_ships;

    unsigned short **_hits;

    int _state;

    char _my_username[MAXLINE];

    char _other_username[MAXLINE];

    NetworkContext *_network_context;

    Move _move;

    Answer _answer;

    bool _ships_spaced;

    bool _username_received;

public:

    explicit Game(QWidget *parent = nullptr);

    void receive_username(char username[MAXLINE]);

    void set_ship(int str1, int str2, int column1, int column2);
```

```
    void make_move(int line, int column);

    bool is_valid_move(int line, int column);

    bool receive_move();

    bool receive_username();

    void process_move();

    void set_ships_spaced(bool ships_spaced);

    bool is_it_possible_to_put_the_ship(int str1, int str2,
                                        int column1, int column2);


    int state() const;

    unsigned short **ships() const;

    bool ships_spaced() const;

    bool username_received() const;

signals:

    void got_username(char username[MAXLINE]);

    void ships_changed(unsigned short **matrix);

    void hits_changed(unsigned short **matrix);

    void state_changed(int state);

    void disconnect_cells(bool flag);
};

#endif // GAME_H
```

Файл «mainwindow.h»:

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include<QMessageBox>
#include <QString>
#include <QByteArray>
#include <QTimer>
#include <field.h>
#include <game.h>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
```

```
    Q_OBJECT


public:

    struct CountShips
    {
        int decks_4;
        int decks_3;
        int decks_2;
        int decks_1;
    };

    MainWindow(QWidget *parent = nullptr);

    void reset_coordinates();

    void setting_ship(int line, int column);

    ~MainWindow();

private slots:
    void on_pushButton_send_username_clicked();

    void _on_get_coordinates(int line, int column);

    void _on_set_username(char username[MAXLINE]);

    void _on_ships_changed(unsigned short **matrix);

    void _on_hits_changed(unsigned short ** matrix);

    void _on_state_changed(int state);

    void _on_disconnect_cells_hits(bool flag);

    void _on_timer_tick();

    void on_toolButton_clicked();

private:

    Ui::MainWindow *ui;

    Game *_game;

    Move _start_pos;

    Move _finish_pos;

    CountShips _count_ships;

    QTimer *_timer;

protected:
    void keyPressEvent(QKeyEvent *event);
};
#endif // MAINWINDOW_H
```

Файл «_game.c»:

```
    #include "_game.h"

    unsigned short **_create_matrix()
    {
        unsigned short **matrix = (unsigned short**) malloc(10 * sizeof
(unsigned short *));

        for(int  i = 0; i < 10; ++i)
            matrix[i] = (unsigned short *) malloc (10 * sizeof (unsigned
short));

        for(int i = 0; i < 10; ++i)
            for(int j = 0; j < 10; ++j)
                matrix[i][j] = Empty;

        return matrix;
    }

    NetworkContext *_network_init(uint16_t port)
    {
        NetworkContext *ctx =  (NetworkContext *)malloc (sizeof
(NetworkContext));

        int sock = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP);
        if(sock < 0){
            perror("socket creation failed");
        }

        ctx->sockfd = sock;
        memset(&ctx->servaddr, 0, sizeof(ctx->servaddr));

        ctx->servaddr.sin_family = AF_INET;
        ctx->servaddr.sin_addr.s_addr = htonl(INADDR_LOOPBACK);
        ctx->servaddr.sin_port = htons(port);

        return ctx;
    }

    bool _is_any_live_ship(unsigned short **matrix)
    {
        for (int i = 0; i < 10; ++i)
            for(int j = 0; j < 10; ++j)
                if(matrix[i][j] == Engaged)
                    return true;

        return false;
    }

    void _change_around_killed(unsigned short **matrix, int line, int
column,
                               int processing_mode)
    {
        switch (processing_mode) {

        case Horizontal:
            if(column - 1 >= 0)
                if(matrix[line][column - 1] == Empty)
```

```
                matrix[line][column - 1] = Neighbor;

        if(column + 1 < 10)
            if(matrix[line][column + 1] == Empty)
                matrix[line][column + 1] = Neighbor;
        break;

    case Vertical:
        if(line - 1 >= 0)
            if(matrix[line - 1][column] == Empty)
                matrix[line - 1][column] = Neighbor;

        if(line + 1 < 10)
            if(matrix[line + 1][column] == Empty)
                matrix[line + 1][column] = Neighbor;
        break;

    case Circular_down:
        if(column - 1 >= 0){
            if(matrix[line][column - 1] == Empty)
                matrix[line][column - 1] = Neighbor;

            if(line + 1 < 10)
                if(matrix[line + 1][column - 1] == Empty)
                    matrix[line + 1][column - 1] = Neighbor;
        }

        if(column + 1 < 10){
            if(matrix[line][column + 1] == Empty)
                matrix[line][column + 1] = Neighbor;

            if(line + 1 < 10)
                if(matrix[line + 1][column + 1] == Empty)
                    matrix[line + 1][column + 1] = Neighbor;
        }

        if(line + 1 < 10)
            if(matrix[line + 1][column] == Empty)
                matrix[line + 1][column] = Neighbor;
        break;

    case Circular_top:
        if(column - 1 >= 0){
            if(matrix[line][column - 1] == Empty)
                matrix[line][column - 1] = Neighbor;

            if(line - 1 >= 0)
                if(matrix[line - 1][column - 1] == Empty)
                    matrix[line - 1][column - 1] = Neighbor;
        }

        if(column + 1 < 10){
            if(matrix[line][column + 1] == Empty)
                matrix[line][column + 1] = Neighbor;

            if(line - 1 >= 0)
                if(matrix[line - 1][column + 1] == Empty)
                    matrix[line - 1][column + 1] = Neighbor;
        }
```

```
            if(line - 1 >= 0)
                if(matrix[line - 1][column] == Empty)
                    matrix[line - 1][column] = Neighbor;
            break;

        case Circular_left:
            if(line - 1 >= 0){
                if(matrix[line - 1][column] == Empty)
                    matrix[line - 1][column] = Neighbor;

                if(column - 1 >= 0)
                    if(matrix[line - 1][column - 1] == Empty)
                        matrix[line - 1][column - 1] = Neighbor;
            }

            if(line + 1 < 10){
                if(matrix[line + 1][column] == Empty)
                    matrix[line + 1][column] = Neighbor;

                if(column - 1 >= 0)
                    if(matrix[line + 1][column - 1] == Empty)
                        matrix[line + 1][column - 1] = Neighbor;
            }

            if(column - 1 >= 0)
                if(matrix[line][column - 1] == Empty)
                    matrix[line][column - 1] = Neighbor;
            break;

        case Circular_right:
            if(line - 1 >= 0){
                if(matrix[line - 1][column] == Empty)
                    matrix[line - 1][column] = Neighbor;

                if(column + 1 < 10)
                    if(matrix[line - 1][column + 1] == Empty)
                        matrix[line - 1][column + 1] = Neighbor;
            }

            if(line + 1 < 10){
                if(matrix[line + 1][column] == Empty)
                    matrix[line + 1][column] = Neighbor;

                if(column + 1 < 10)
                    if(matrix[line + 1][column + 1] == Empty)
                        matrix[line + 1][column + 1] = Neighbor;
            }

            if(column + 1 < 10)
                if(matrix[line][column + 1] == Empty)
                    matrix[line][column + 1] = Neighbor;
            break;
        }
    }
    bool _is_horizontal(unsigned short **matrix, int line, int column)
    {
        if(column - 1 >= 0)
            if(matrix[line][column - 1] == Hit || matrix[line][column -
1] == Killed)
                return true;
```

```
        if(column + 1 < 10)
            if(matrix[line][column + 1] == Hit || matrix[line][column +
1] == Killed)
                return true;

        return false;
    }

    void _change_after_kill(unsigned short **matrix, int line, int
column)
    {
        if(_is_horizontal (matrix, line, column)){

            _change_around_killed (matrix, line, column, Vertical);

            bool flag_left = true;
            bool flag_right = true;

            if(column - 1 >= 0){
                if (matrix[line][column - 1] == Hit ||
matrix[line][column - 1] == Killed){
                    matrix[line][column - 1] = Killed;
                    _change_around_killed (matrix, line, column - 1,
Vertical);

                }else{

                    flag_left = false;
                    _change_around_killed (matrix, line, column,
Circular_left);
                }

                if(column - 2 >= 0 && flag_left){
                    if(matrix[line][column - 2] == Hit ||
matrix[line][column - 2] == Killed){
                        matrix[line][column - 2] = Killed;
                        _change_around_killed (matrix, line, column - 2,
Vertical);

                    }else{

                        flag_left = false;
                        _change_around_killed (matrix, line, column - 1,
Circular_left);
                    }

                    if(column - 3 >= 0 && flag_left){
                        if(matrix[line][column - 3] == Hit||
matrix[line][column - 3] == Killed){
                            matrix[line][column - 3] = Killed;
                            _change_around_killed (matrix, line, column -
3, Circular_left);

                        }else

                            _change_around_killed (matrix, line, column -
2, Circular_left);
                    }
                }
```

```
            }

            if(column + 1 < 10){

                if(matrix[line][column + 1] == Hit|| matrix[line][column
+ 1] == Killed){
                    matrix[line][column + 1] = Killed;
                    _change_around_killed (matrix, line, column + 1,
Vertical);

                }else{

                    flag_right = false;
                    _change_around_killed (matrix, line, column,
Circular_right);
                }

                if(column + 2 < 10 && flag_right){
                    if(matrix[line][column + 2] == Hit||
matrix[line][column + 2] == Killed){
                        matrix[line][column + 2] = Killed;
                        _change_around_killed (matrix, line, column + 2,
Vertical);

                    }else{

                        flag_right = false;
                        _change_around_killed (matrix, line, column + 1,
Circular_right);
                    }

                    if(column + 3 < 10 && flag_right){
                        if(matrix[line][column + 3] == Hit||
matrix[line][column + 3] == Killed){
                            matrix[line][column + 3] = Killed;
                            _change_around_killed (matrix, line, column +
3, Circular_right);

                        }else

                            _change_around_killed (matrix, line, column +
2, Circular_right);
                    }
                }
            }
        }

        if(!_is_horizontal (matrix, line, column)){

            bool flag_top = true;
            bool flag_down = true;

            _change_around_killed (matrix, line, column, Horizontal);

            if(line - 1 >= 0){
                if(matrix[line - 1][column] == Hit|| matrix[line -
1][column] == Killed){
                    matrix[line - 1][column] = Killed;
                    _change_around_killed (matrix, line - 1, column,
Horizontal);
```

```
                }else{

                    flag_top = false;
                    _change_around_killed (matrix, line, column,
Circular_top);
                }

                if(line - 2 >= 0 && flag_top){
                    if(matrix[line - 2][column] == Hit|| matrix[line -
2][column] == Killed){
                        matrix[line - 2][column] = Killed;
                        _change_around_killed (matrix, line - 2, column,
Horizontal);

                    }else{

                        flag_top = false;
                        _change_around_killed (matrix, line - 1, column,
Circular_top);
                    }

                    if(line - 3 >= 0 && flag_top){
                        if(matrix[line - 3][column] == Hit|| matrix[line
- 3][column] == Killed){
                            matrix[line - 3][column] = Killed;
                            _change_around_killed (matrix, line - 3,
column, Circular_top);

                        }else

                            _change_around_killed (matrix, line - 2,
column, Circular_top);
                    }
                }
            }
            if(line + 1 < 10){

                if(matrix[line + 1][column] == Hit|| matrix[line +
1][column] == Killed){
                    matrix[line + 1][column] = Killed;
                    _change_around_killed (matrix, line + 1, column,
Horizontal);

                }else{

                    _change_around_killed (matrix, line, column,
Circular_down);
                    flag_down = false;
                }

                if(line + 2 < 10 && flag_down){
                    if(matrix[line + 2][column] == Hit|| matrix[line +
2][column] == Killed){
                        matrix[line + 2][column] = Killed;
                        _change_around_killed (matrix, line + 2, column,
Horizontal);

                    }else{
```

```
                        _change_around_killed (matrix, line + 1, column,
Circular_down);
                        flag_down = false;
                    }

                    if(line + 3 < 10 && flag_down){
                        if(matrix[line + 3][column] == Hit|| matrix[line
+ 3][column] == Killed){
                            matrix[line + 3][column] = Killed;
                            _change_around_killed (matrix, line + 3,
column, Circular_down);

                        }else

                            _change_around_killed (matrix, line + 2,
column, Circular_down);
                    }
                }
            }
        }
    }

    bool _is_killed(unsigned short **matrix, int line, int column)
    {
        bool flag_left = true;
        bool flag_right = true;
        bool flag_top = true;
        bool flag_down = true;

        if(column - 1 >= 0){
            if(matrix[line][column - 1] == Engaged)
                return false;

            else if (matrix[line][column - 1] != Hit)
                flag_left = false;

            if(column - 2 >= 0 && flag_left){
                if(matrix[line][column - 2] == Engaged)
                    return false;

                else if(matrix[line][column - 2] != Hit)
                    flag_left = false;
            }

            if(column - 3 >= 0 && flag_left){
                if(matrix[line][column - 3] == Engaged)
                    return false;
            }
        }

        if(column + 1 < 10){

            if(matrix[line][column + 1] == Engaged)
                return false;

            else if (matrix[line][column + 1] != Hit)
                flag_right = false;

            if(column + 2 < 10 && flag_right){
                if(matrix[line][column + 2] == Engaged)
```

```
                return false;

            else if(matrix[line][column + 2] != Hit)
                flag_right = false;

            if(column + 3 < 10 && flag_right){
                if(matrix[line][column + 3] == Engaged)
                    return false;
            }
        }
    }

    if(line - 1 >= 0){

        if(matrix[line - 1][column] == Engaged)
            return false;

        else if (matrix[line - 1][column] != Hit)
            flag_top = false;

        if(line - 2 >= 0 && flag_top){
            if(matrix[line - 2][column] == Engaged)
                return false;

            else if(matrix[line - 2][column] != Hit)
                flag_top = false;

            if(line - 49 - 3 >= 0 && flag_top){
                if(matrix[line - 3][column] == Engaged)
                    return false;
            }
        }
    }

    if(line + 1 < 10){

        if(matrix[line + 1][column] == Engaged)
            return false;

        else if(matrix[line + 1][column] != Hit)
            flag_down = false;

        if(line + 2 < 10 && flag_down){
            if(matrix[line + 2][column] == Engaged)
                return false;

            else if(matrix[line + 2][column] != Hit)
                flag_down = false;

            if(line + 3 < 10 && flag_down){
                if(matrix[line + 3][column] == Engaged)
                    return false;
            }
        }
    }
    return true;

}

int _result_of_shot (unsigned short **matrix, int line, int column)
```

```
    {
        if(matrix[line][column] == Engaged && _is_killed (matrix, line,
column))
            return Killed;

        if(matrix[line][column] == Engaged && !_is_killed (matrix, line,
column))
            return Hit;

        if(matrix[line][column] == Empty ||
                matrix[line][column] == Neighbor)
            return Miss;

        return ERROR;
    }

    void _set_ship(unsigned short **matrix, int str1, int str2, int
column1, int column2)
    {
        for (int i = str1; i <= str2; ++i){
            for(int j = column1; j <= column2; ++j){

                matrix[i][j] = Engaged;

                if (i - 1 >= 0 && str1 == str2)
                    matrix[i - 1][j] = Neighbor;

                if (i + 1 <= 9 && str1 == str2)
                    matrix[i + 1][j] = Neighbor;

                if (j - 1 >= 0 && column1 == column2)
                    matrix[i][j - 1] = Neighbor;

                if (j + 1 <= 9 && column1 == column2)
                    matrix[i][j + 1] = Neighbor;
            }
        }

        if(column1 - 1 >= 0 && str1 == str2){

            matrix[str1][column1 - 1] = Neighbor;

            if(str1 - 1 >= 0)
                matrix[str1 - 1][column1 - 1] = Neighbor;

            if(str1 + 1 <= 9 )
                matrix[str1 + 1][column1 - 1] = Neighbor;
        }

        if(column2 + 1 <= 9 && str1 == str2){

            matrix[str1][column2 + 1] = Neighbor;

            if(str1 - 1 >= 0)
                matrix[str1 - 1][column2 + 1] = Neighbor;

            if(str1 + 1 <= 9 )
                matrix[str1 + 1][column2 + 1] = Neighbor;
        }
```

```
        if(str1 - 1 >= 0 && column1 == column2){

            matrix[str1 - 1][column1] = Neighbor;

            if(column1 - 1 >= 0)
                matrix[str1 - 1][column1 - 1] = Neighbor;

            if(column1 + 1 <= 9 )
                matrix[str1 - 1][column1 + 1] = Neighbor;
        }

        if(str2 + 1 <= 9 && column1 == column2){

            matrix[str2 + 1][column1] = Neighbor;

            if(column1 - 1 >= 0)
                matrix[str2 + 1][column1 - 1] = Neighbor;

            if(column1 + 1 <= 9)
                matrix[str2 + 1][column1 + 1] = Neighbor;
        }
    }


    bool _is_it_possible_to_put_the_ship(unsigned short **matrix, int
str1, int str2,
                                         int column1, int column2)
    {
        if (str1 > 9 || str1 < 0 || str2 > 9 || str2 < 0 || str1 > str2)
            return false;

        if (column1 > 9 || column1 < 0 || column2 > 9 || column2 < 0 ||
column1 > column2)
            return false;

        if(str1 != str2 && column1 != column2)
            return false;

        for (int i = str1; i <= str2; ++i)
            for(int j = column1; j <= column2; ++j)
                if (matrix[i][j] != Empty)
                    return false;

        return true;
    }

    bool _is_valid_move(unsigned short **matrix, int line, int column)
    {
        if(matrix[line][column] != Empty)
            return false;

        return true;
    }

    void send_move(NetworkContext *ctx, Move move)
    {
        sendto(ctx->sockfd,
               &move,
               sizeof (Move),
               0,
```

```
            (const struct sockaddr *) &ctx->servaddr,
            sizeof(ctx->servaddr));
}

void send_answer(NetworkContext *ctx, Answer answer)
{
    sendto(ctx->sockfd,
           &answer,
           sizeof (Answer),
           0,
           (const struct sockaddr *) &ctx->servaddr,
           sizeof(ctx->servaddr));
}

void _send_username(NetworkContext *ctx, char username[])
{
    sendto(ctx->sockfd,
           (const char *) username,
           strlen(username),
           0,
           (const struct sockaddr *) &ctx->servaddr,
           sizeof(ctx->servaddr));
}

int receive_confirm(NetworkContext *ctx)
{
    int res = 0;
    int len = sizeof (ctx->servaddr);

    recvfrom (ctx->sockfd,
              &res,
              sizeof (Answer),
              MSG_WAITALL,
              (struct sockaddr *) &ctx->servaddr,
              (socklen_t *)&len);

    return res;
}

Answer receive_answer(NetworkContext *ctx)
{
    Answer answer;
    int len = sizeof (ctx->servaddr);

    recvfrom (ctx->sockfd,
              &answer,
              sizeof (Answer),
              MSG_WAITALL,
              (struct sockaddr *) &ctx->servaddr,
              (socklen_t *)&len);

    return answer;
}

Move _receive_move(NetworkContext *ctx)
{
    Move move;
    int len = sizeof (ctx->servaddr);

    int result = recvfrom (ctx->sockfd,
```

```
                                &move,
                                sizeof (Move),
                                MSG_DONTWAIT,
                                (struct sockaddr *) &ctx->servaddr,
                                (socklen_t *)&len);

    if(result < 0){
        move.line = -1;
        move.column = -1;
    }

    return move;
}

int receive_state(NetworkContext *ctx)
{
    int state;
    int len = sizeof (ctx->servaddr);

    recvfrom (ctx->sockfd,
              &state,
              sizeof (int),
              MSG_WAITALL,
              (struct sockaddr *) &ctx->servaddr,
              (socklen_t *)&len);
    return state;
}

 void _receive_username(NetworkContext *ctx, char username[MAXLINE])
{
    int len = sizeof (ctx->servaddr);

    int res  = recvfrom (ctx->sockfd,
              (char *)username,
              MAXLINE,
              MSG_DONTWAIT,
              (struct sockaddr *) &ctx->servaddr,
              (socklen_t *)&len);

    if(res < 0)
        strcpy (username, "");
}
```

Файл «cell.cpp»:

```
#include "cell.h"

Cell::Cell(QObject *parent):
    _line(-1),
    _column(-1)
{
}

void Cell::setCoordinates(int line, int column)
{
    _line = line;
    _column = column;
```

```
}

void Cell::mousePressEvent(QMouseEvent *event)
{
    emit cell_clicked (_line, _column);
}
```

Файл «field.cpp»:

```
#include "field.h"
#include <game.h>

void Field::get_coordinates(int line, int column)
{
    emit send_move (line, column);
}

Field::Field(QWidget *parent) : QWidget(parent)
{
    QGridLayout *gridLayout = new QGridLayout(this);

    for(int i = 1; i < 11; ++i){

        QLabel *label = new QLabel(this);

        label->setText (QString("%1").arg (i - 1));

        label->setMinimumSize (100, 100);

        label->setAlignment (Qt::AlignCenter);

        gridLayout->addWidget (label, i , 0, 1, 1,Qt::AlignCenter);
    }

    for(int i = 1; i < 11; ++i){

        QLabel *label = new QLabel(this);

        char c = i + 64;

        label->setText (QString("%1").arg (c));

        label->setMinimumSize (100, 100);

        label->setAlignment (Qt::AlignCenter);

        gridLayout->addWidget (label, 0, i, 1, 1, Qt::AlignCenter);
    }

    for (int i =  1; i < FIELD_LENGTH + 1; ++i ) {
        for (int j = 1; j < FIELD_WIDTH + 1; ++j) {

            Cell *cell = new Cell(this);

            cell->setStyleSheet ("background-color: green");

            cell->setCoordinates (i - 1, j - 1);
```

```
                cell->setMinimumSize (100, 100);

                gridLayout->addWidget(cell, i + 1,  j + 1, 1, 1);

                _cells[i - 1][j - 1] = cell;

                connect (_cells[i - 1][j - 1],
SIGNAL(cell_clicked(int,int)),
                         this, SLOT(get_coordinates(int,int)));
            }
        }
    }


    void Field::update_matrix(unsigned short **matrix)
    {
        for (int i = 0; i < FIELD_LENGTH; ++i) {
            for(int j = 0; j < FIELD_WIDTH; ++j){

                if(matrix[i][j] == Empty)
                    _cells[i][j]->setStyleSheet ("background-color:
green");

                else if(matrix[i][j] == Neighbor)
                    _cells[i][j]->setStyleSheet ("background-color:
yellow");

                else if(matrix[i][j] == Engaged)
                    _cells[i][j]->setStyleSheet ("background-color:
blue");

                else if(matrix[i][j] == Miss)
                    _cells[i][j]->setStyleSheet ("background-color:
red");

                else if(matrix[i][j] == Hit)
                    _cells[i][j]->setStyleSheet ("background-color:
purple");

                else if(matrix[i][j] == Killed)
                    _cells[i][j]->setStyleSheet ("background-color:
rgb(100, 0, 0)");

            }
        }
    }
```

Файл «game.cpp»:

```
    #include "game.h"
    #include<_game.h>

    Game::Game(QWidget *parent):
        QWidget (parent)
        ,_state(Unknown)
        ,_my_username("")
        ,_other_username("")
```

```
        ,_ships_spaced(false)
        ,_username_received(false)
    {
        _move.column = -1;
        _move.line = -1;

        _answer.is_any_live_ship = true;
        _answer.result_of_shoot = -1;

        _ships = _create_matrix ();

        _hits = _create_matrix ();

        _network_context = _network_init (PORT);
    }

    bool Game::username_received() const
    {
        return _username_received;
    }

    bool Game::ships_spaced() const
    {
        return _ships_spaced;
    }

    bool Game::is_it_possible_to_put_the_ship( int str1,
                                               int str2, int column1, int
column2)
    {
        return _is_it_possible_to_put_the_ship(_ships, str1, str2,
column1, column2);
    }

    bool Game::is_valid_move(int line, int column)
    {
        return _is_valid_move(_hits, line, column);
    }

    unsigned short **Game::ships() const
    {
        return _ships;
    }

    bool Game::receive_move()
    {
        _move = _receive_move(_network_context);
        return !(_move.line < 0 || _move.column < 0);
    }

    bool Game::receive_username()
    {
        _receive_username (_network_context, _other_username);

        if(strcmp (_other_username, "") == 0)
            return false;

        emit got_username (_other_username);
        _username_received = true;
```

```
        return true;
    }

    void Game::process_move()
    {
        _answer.result_of_shoot = _result_of_shot (_ships, _move.line,
_move.column);

        _answer.is_any_live_ship = true;

        _ships[_move.line][_move.column] = _answer.result_of_shoot;

        if(_answer.result_of_shoot == Killed){
            _change_after_kill (_ships, _move.line, _move.column);
            _answer.is_any_live_ship = _is_any_live_ship (_ships);
        }

        emit ships_changed (_ships);

        send_answer(_network_context, _answer);

        if(_answer.result_of_shoot != Hit && _answer.result_of_shoot !=
Killed){
            _state = Making_move;
            emit state_changed (_state);
        }

        if (!_is_any_live_ship (_ships)){
            _state = Loser;
            emit state_changed (_state);
        }

    }

    void Game::set_ships_spaced(bool ships_spaced){_ships_spaced =
ships_spaced;}

    int Game::state() const{return _state;}

    void Game::receive_username(char username[MAXLINE])
    {
        strcpy (_my_username, username);

        _send_username(_network_context, _my_username);

        _state = receive_state(_network_context);

        emit state_changed (_state);

    }

    void Game::set_ship(int str1, int str2, int column1, int column2)
    {_set_ship (_ships, str1, str2, column1, column2);}

    void Game::make_move(int line, int column)
    {
        _move.line = line;
        _move.column = column;

        while (true){
```

```
            send_move(_network_context, _move);

            if (receive_confirm(_network_context) > 0)
                break;

        }

        _answer = receive_answer(_network_context);

        _hits[_move.line][_move.column ] = _answer.result_of_shoot;

        if (_answer.result_of_shoot == Killed){
            _change_after_kill (_hits, _move.line, _move.column);

            if(_answer.is_any_live_ship == 0){
                _state = Winner;
                emit state_changed (_state);
            }
        }

        emit hits_changed (_hits);

        if(_answer.result_of_shoot != Hit && _answer.result_of_shoot !=
Killed){
            _state = Waiting_move;
            emit state_changed (_state);
        }
    }
```

Файл «main.cpp»:

```
    #include "mainwindow.h"

    #include <QApplication>

    int main(int argc, char *argv[])
    {
        QApplication a(argc, argv);
        MainWindow w;
        w.show();
        return a.exec();
    }
```

Файл «mainwindow.cpp»:

```
    #include "mainwindow.h"
    #include "ui_mainwindow.h"
    #include<QKeyEvent>

    MainWindow::MainWindow(QWidget *parent)
        : QMainWindow(parent)
        , ui(new Ui::MainWindow)
        ,_game(new Game())
        ,_timer (new QTimer(this))
    {
```

```
        _start_pos.line = -1;
        _start_pos.column = -1;

        _finish_pos.line = -1;
        _finish_pos.column = -1;

        _count_ships.decks_1 = 4;
        _count_ships.decks_2 = 3;

        _count_ships.decks_3 = 2;
        _count_ships.decks_4 = 1;

        ui->setupUi(this);
        _timer->setInterval(1);
        _timer->start();

        connect(_timer, SIGNAL(timeout()), this, SLOT(_on_timer_tick()));

        connect (ui->field_ships, SIGNAL(send_move(int,int)),
                 this, SLOT(_on_get_coordinates(int,int)));

        connect (_game, SIGNAL(state_changed(int)), this,
                 SLOT(_on_state_changed(int)));

        connect (_game, &Game::got_username,
                 this, &MainWindow::_on_set_username);

        connect (_game, SIGNAL(ships_changed(unsigned short**)), this,
                 SLOT(_on_ships_changed(unsigned short**)));

        connect (_game, SIGNAL(hits_changed(unsigned short**)), this,
                 SLOT(_on_hits_changed(unsigned short**)));

        QMessageBox :: information(this,
                                   QString("Rules"),
                                   QString("Before sending the username,
you must arrange all "
     "the ships (1 - four-decker, 2 - three-decks, 3 - double-decks , 4 -
single-decks)."
     " To install a ship, click on the field that is the beginning of the
ship, and then "
     "on the field that is the end of the ship. If you want to reset the
start coordinate,"
     " press the Esk key. If you want to send the username, press the
Enter key."));
     }

     void MainWindow::reset_coordinates()
     {
         _start_pos.line = -1;
         _start_pos.column = -1;
         _finish_pos.line = -1;
         _finish_pos.column = -1;
     }

     void MainWindow::setting_ship(int line, int column)
     {
         if(_start_pos.line == -1 && _start_pos.column == -1){
             _start_pos.line = line;
             _start_pos.column = column;
```

```
        } else if(_start_pos.line != -1 && _start_pos.column != -1){

            _finish_pos.line = line;
            _finish_pos.column = column;

            if (_finish_pos.column < _start_pos.column){
                int tmp = _finish_pos.column;
                _finish_pos.column = _start_pos.column;
                _start_pos.column = tmp;
            }

            if(_finish_pos.line < _start_pos.line){
                int tmp = _finish_pos.line;
                _finish_pos.line = _start_pos.line;
                _start_pos.line = tmp;
            }

            if(!_game->is_it_possible_to_put_the_ship (_start_pos.line,
_finish_pos.line,
                                                       _start_pos.column,
_finish_pos.column)){

                QMessageBox::warning (this, "Error",
                                      "You enter incorrect coordinates,
try again");

                reset_coordinates();

            }else if(_finish_pos.line - _start_pos.line == 3
                     || _finish_pos.column - _start_pos.column == 3){

                if(_count_ships.decks_4 > 0){

                    _game->set_ship (_start_pos.line, _finish_pos.line,
                                     _start_pos.column,
_finish_pos.column);
                    ui->label_four_decker->setText
                            (QString("Four - decker %1/1").arg (2 -
_count_ships.decks_4));
                    ui->field_ships->update_matrix (_game->ships ());

                    --_count_ships.decks_4;

                    if(_count_ships.decks_4 == 0){
                        QFont font;
                        font.setStrikeOut(true);
                        ui->label_four_decker->setFont (font);
                    }
                    reset_coordinates();

                } else {

                    QMessageBox::warning (this, "Error",
                                          "You have installed enough
ships with lenght 4, "
                                              "choose another ship");

                    reset_coordinates();
                }
```

```
            } else if(_finish_pos.line - _start_pos.line == 2
                      || _finish_pos.column - _start_pos.column == 2){

                if(_count_ships.decks_3 > 0){

                    _game->set_ship (_start_pos.line, _finish_pos.line,
                                     _start_pos.column,
_finish_pos.column);

                    ui->label_three_deck->setText
                            (QString("Three - deck %1/2").arg (3 -
_count_ships.decks_3));
                    ui->field_ships->update_matrix (_game->ships ());

                    --_count_ships.decks_3;

                    if(_count_ships.decks_3 == 0){
                        QFont font;
                        font.setStrikeOut(true);
                        ui->label_three_deck->setFont (font);
                    }
                    reset_coordinates();

                } else {

                    QMessageBox::warning (this, "Error",
                                          "You have installed enough
ships with lenght 3, "
                                              "choose another ship");

                    reset_coordinates();
                }

            } else if(_finish_pos.line - _start_pos.line == 1
                      || _finish_pos.column - _start_pos.column == 1){

                if(_count_ships.decks_2 > 0){

                    _game->set_ship (_start_pos.line, _finish_pos.line,
                                     _start_pos.column,
_finish_pos.column);
                    ui->label_double_decker->setText
                            (QString("Double - decker %1/3").arg (4 -
_count_ships.decks_2));
                    ui->field_ships->update_matrix (_game->ships ());

                    --_count_ships.decks_2;

                    if(_count_ships.decks_2 == 0){
                        QFont font;
                        font.setStrikeOut(true);
                        ui->label_double_decker->setFont (font);
                    }

                    reset_coordinates();

                } else {

                    QMessageBox::warning (this, "Error",
```

```
                                        "You have installed enough
ships with lenght 2, "
                                            "choose another ship");

                    reset_coordinates();
                }

            } else if(_finish_pos.line - _start_pos.line == 0
                      && _finish_pos.column - _start_pos.column == 0){

                if(_count_ships.decks_1 > 0){

                    _game->set_ship (_start_pos.line, _finish_pos.line,
                                     _start_pos.column,
_finish_pos.column);

                    ui->label_single_deck->setText
                            (QString("Single - deck %1/4").arg (5 -
_count_ships.decks_1));

                    ui->field_ships->update_matrix (_game->ships ());

                    --_count_ships.decks_1;

                    if(_count_ships.decks_1 == 0){
                        QFont font;
                        font.setStrikeOut(true);
                        ui->label_single_deck->setFont (font);
                    }

                    reset_coordinates();

                } else {

                    QMessageBox::warning (this, "Error",
                                        "You have installed enough
ships with lenght 1, "
                                            "choose another ship");

                    reset_coordinates();
                }

            } else {

                QMessageBox::warning (this, "Error",
                                      "You try to set ship with len more
then 4, "
                                        "choose another ship");

                reset_coordinates();
            }
        }

        if (_count_ships.decks_4 == 0 && _count_ships.decks_3 == 0 &&
                _count_ships.decks_2 == 0 && _count_ships.decks_1 == 0){

            _game->set_ships_spaced (true);

            disconnect (ui->field_ships, SIGNAL(send_move(int,int)),
                        this, SLOT(_on_get_coordinates(int,int)));
```

```
            ui->label_state->setText ("Ready to send username");
        }
    }

    MainWindow::~MainWindow()
    {
        delete ui;
        delete _game;
    }

    void MainWindow::on_pushButton_send_username_clicked()
    {
        if(_game->ships_spaced ()){

            ui->lineEdit->setEnabled (false);

            ui->pushButton_send_username->setEnabled (false);

            QString username = ui->lineEdit->text ();
            QByteArray ba = username.toLocal8Bit ();

            _game->receive_username (ba.data ());
        } else
            QMessageBox :: warning(this, "Error",
                                   "You must set all of your ships before
send username");
    }

    void MainWindow::_on_get_coordinates(int line, int column)
    {
        if(!_game->ships_spaced ())
            setting_ship (line, column);

        else if(_game->is_valid_move (line, column))
            _game->make_move (line, column);

        else if(!_game->is_valid_move (line, column))
            QMessageBox::warning (this, "Error",
                                  "You try to shoot in field which is not
empty ");
    }

    void MainWindow::_on_set_username(char username[MAXLINE])
    {
        ui->label_other_username->
                setText (QString("Opponent username: %1").arg
(username));

        _on_disconnect_cells_hits (false);
    }

    void MainWindow::_on_ships_changed(unsigned short **matrix)
    {
        ui->field_ships->update_matrix (matrix);
    }

    void MainWindow::_on_hits_changed(unsigned short **matrix)
    {
        ui->field_hits->update_matrix (matrix);
    }
```

```
void MainWindow::_on_state_changed(int state)
{
    if (state == Making_move){
        ui->label_state->setText ("Making move");

        if(_game->username_received ())
            _on_disconnect_cells_hits (false);
    }

    if (state == Waiting_move){
        ui->label_state->setText ("Waiting move");
        _on_disconnect_cells_hits (true);
    }

    if (state == Winner){
        QMessageBox::information (this,
                                  QString("Congratulations"),
                                  QString("You won this match!!!"));

        ui->label_state->setText ("Winner");
        _on_disconnect_cells_hits (true);
    }

    if (state == Loser){
        QMessageBox::information (this,
                                  QString("Result of match"),
                                  QString("You lose this match!!!"));

        ui->label_state->setText ("Loser");
        _on_disconnect_cells_hits (true);
    }
}

void MainWindow::_on_disconnect_cells_hits(bool flag)
{
    if(!flag)
        connect (ui->field_hits, SIGNAL(send_move(int,int)),
                 this, SLOT(_on_get_coordinates(int,int)));

    else
        disconnect (ui->field_hits, SIGNAL(send_move(int,int)),
                    this, SLOT(_on_get_coordinates(int,int)));
}

void MainWindow::_on_timer_tick()
{
    if(!_game->username_received ())
        _game->receive_username ();

    else if(_game->state() == Waiting_move)
        if(_game->receive_move ())
            _game->process_move ();
}

void MainWindow::keyPressEvent(QKeyEvent *event)
{
    if(event->key ()== Qt::Key_Escape){
        if(!_game->ships_spaced ())
            reset_coordinates ();
```

```
        }else if(event->key() == Qt::Key_Return){

            if(_game->ships_spaced ()){

                ui->lineEdit->setEnabled (false);

                ui->pushButton_send_username->setEnabled (false);

                QString username = ui->lineEdit->text ();
                QByteArray ba = username.toLocal8Bit ();

                _game->receive_username (ba.data ());
            } else
                QMessageBox :: warning(this, "Error",
                                       "You must set all of your ships
before send username");
        }
    }

    void MainWindow::on_toolButton_clicked()
    {
        QMessageBox :: information(this,
                                   QString("Rules"),
                                   QString("Before sending the username,
you must arrange all "
    "the ships (1 - four-decker, 2 - three-decks, 3 - double-decks , 4 -
single-decks)."
    " To install a ship, click on the field that is the beginning of the
ship, and then "
    "on the field that is the end of the ship. If you want to reset the
start coordinate,"
    " press the Esk key. If you want to send the username, press the
Enter key."));
    }
```

Файл «mainwindow.ui»:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <ui version="4.0">
     <class>MainWindow</class>
     <widget class="QMainWindow" name="MainWindow">
      <property name="geometry">
       <rect>
        <x>0</x>
        <y>0</y>
        <width>1386</width>
        <height>578</height>
       </rect>
      </property>
      <property name="windowTitle">
       <string>MainWindow</string>
      </property>
      <widget class="QWidget" name="centralwidget">
       <widget class="QGroupBox" name="groupBox">
        <property name="geometry">
         <rect>
```

```
            <x>9</x>
            <y>9</y>
            <width>1321</width>
            <height>521</height>
           </rect>
          </property>
          <property name="title">
           <string>GroupBox</string>
          </property>
          <layout class="QHBoxLayout" name="horizontalLayout">
           <item>
            <layout class="QVBoxLayout" name="verticalLayout">
             <item>
              <widget class="QLabel" name="label">
               <property name="text">
                <string>Username:</string>
               </property>
              </widget>
             </item>
             <item>
              <widget class="QLineEdit" name="lineEdit"/>
             </item>
             <item>
              <layout class="QHBoxLayout" name="horizontalLayout_3">
               <item>
                <widget class="QPushButton"
name="pushButton_send_username">
                 <property name="text">
                  <string>Send username</string>
                 </property>
                </widget>
               </item>
               <item>
                <widget class="QToolButton" name="toolButton">
                 <property name="text">
                  <string>?</string>
                 </property>
                 <property name="toolButtonStyle">
                  <enum>Qt::ToolButtonTextBesideIcon</enum>
                 </property>
                </widget>
               </item>
              </layout>
             </item>
             <item>
              <widget class="QLabel" name="label_other_username">
               <property name="text">
                <string>...</string>
               </property>
               <property name="alignment">
                <set>Qt::AlignCenter</set>
               </property>
              </widget>
             </item>
             <item>
              <widget class="QLabel" name="label_state">
               <property name="text">
                <string>Placement of ships</string>
               </property>
               <property name="alignment">
```

```
    <set>Qt::AlignCenter</set>
   </property>
  </widget>
 </item>
 <item>
  <layout class="QVBoxLayout" name="verticalLayout_2">
   <item>
    <widget class="QLabel" name="label_18">
     <property name="text">
      <string>Spips spaced:</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLabel" name="label_four_decker">
     <property name="text">
      <string>Four - decker 0/1</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLabel" name="label_three_deck">
     <property name="text">
      <string>Three - deck 0/2</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLabel" name="label_double_decker">
     <property name="text">
      <string>Double - decker 0/3</string>
     </property>
    </widget>
   </item>
   <item>
    <widget class="QLabel" name="label_single_deck">
     <property name="text">
      <string>Single - deck 0/4</string>
     </property>
    </widget>
   </item>
  </layout>
 </item>
 <item>
  <layout class="QHBoxLayout" name="horizontalLayout_2">
   <item>
    <layout class="QVBoxLayout" name="verticalLayout_4">
     <item>
      <widget class="QLabel" name="label_3">
       <property name="minimumSize">
        <size>
         <width>25</width>
         <height>25</height>
        </size>
       </property>
       <property name="maximumSize">
        <size>
         <width>25</width>
         <height>25</height>
        </size>
```

```
   </property>
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: green</string>
   </property>
   <property name="text">
    <string/>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QLabel" name="label_5">
   <property name="minimumSize">
    <size>
     <width>25</width>
     <height>25</height>
    </size>
   </property>
   <property name="maximumSize">
    <size>
     <width>25</width>
     <height>25</height>
    </size>
   </property>
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: yellow</string>
   </property>
   <property name="text">
    <string/>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QLabel" name="label_6">
   <property name="minimumSize">
    <size>
     <width>25</width>
     <height>25</height>
    </size>
   </property>
   <property name="maximumSize">
    <size>
     <width>25</width>
     <height>25</height>
    </size>
   </property>
   <property name="palette">
    <palette>
     <active>
      <colorrole role="Button">
       <brush brushstyle="SolidPattern">
        <color alpha="255">
         <red>255</red>
         <green>0</green>
         <blue>0</blue>
```

```
    </color>
   </brush>
  </colorrole>
  <colorrole role="Base">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>0</green>
     <blue>0</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Window">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>0</green>
     <blue>0</blue>
    </color>
   </brush>
  </colorrole>
 </active>
 <inactive>
  <colorrole role="Button">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>0</green>
     <blue>0</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Base">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>0</green>
     <blue>0</blue>
    </color>
   </brush>
  </colorrole>
  <colorrole role="Window">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>0</green>
     <blue>0</blue>
    </color>
   </brush>
  </colorrole>
 </inactive>
 <disabled>
  <colorrole role="Button">
   <brush brushstyle="SolidPattern">
    <color alpha="255">
     <red>255</red>
     <green>0</green>
     <blue>0</blue>
    </color>
   </brush>
```

```
      </colorrole>
      <colorrole role="Base">
       <brush brushstyle="SolidPattern">
        <color alpha="255">
         <red>255</red>
         <green>0</green>
         <blue>0</blue>
        </color>
       </brush>
      </colorrole>
      <colorrole role="Window">
       <brush brushstyle="SolidPattern">
        <color alpha="255">
         <red>255</red>
         <green>0</green>
         <blue>0</blue>
        </color>
       </brush>
      </colorrole>
     </disabled>
    </palette>
   </property>
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: red</string>
   </property>
   <property name="text">
    <string/>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QLabel" name="label_7">
   <property name="minimumSize">
    <size>
     <width>25</width>
     <height>25</height>
    </size>
   </property>
   <property name="maximumSize">
    <size>
     <width>25</width>
     <height>25</height>
    </size>
   </property>
   <property name="autoFillBackground">
    <bool>false</bool>
   </property>
   <property name="styleSheet">
    <string notr="true">background-color: blue</string>
   </property>
   <property name="text">
    <string/>
   </property>
  </widget>
 </item>
 <item>
  <widget class="QLabel" name="label_9">
```

```
                    <property name="minimumSize">
                     <size>
                      <width>25</width>
                      <height>25</height>
                     </size>
                    </property>
                    <property name="maximumSize">
                     <size>
                      <width>25</width>
                      <height>25</height>
                     </size>
                    </property>
                    <property name="autoFillBackground">
                     <bool>false</bool>
                    </property>
                    <property name="styleSheet">
                     <string notr="true">background-color: purple</string>
                    </property>
                    <property name="text">
                     <string/>
                    </property>
                   </widget>
                  </item>
                  <item>
                   <widget class="QLabel" name="label_12">
                    <property name="minimumSize">
                     <size>
                      <width>25</width>
                      <height>25</height>
                     </size>
                    </property>
                    <property name="maximumSize">
                     <size>
                      <width>25</width>
                      <height>25</height>
                     </size>
                    </property>
                    <property name="styleSheet">
                     <string notr="true">background-color: rgb(100, 0,
0)</string>
                    </property>
                    <property name="text">
                     <string/>
                    </property>
                   </widget>
                  </item>
                 </layout>
                </item>
                <item>
                 <layout class="QVBoxLayout" name="verticalLayout_3">
                  <item>
                   <widget class="QLabel" name="label_10">
                    <property name="text">
                     <string>Empty</string>
                    </property>
                    <property name="alignment">
                     <set>Qt::AlignCenter</set>
                    </property>
                   </widget>
                  </item>
```

```
            <item>
             <widget class="QLabel" name="label_11">
              <property name="text">
               <string>Neighbor</string>
              </property>
              <property name="alignment">
               <set>Qt::AlignCenter</set>
              </property>
             </widget>
            </item>
            <item>
             <widget class="QLabel" name="label_13">
              <property name="text">
               <string>Engaged</string>
              </property>
              <property name="alignment">
               <set>Qt::AlignCenter</set>
              </property>
             </widget>
            </item>
            <item>
             <widget class="QLabel" name="label_16">
              <property name="font">
               <font>
                <strikeout>false</strikeout>
               </font>
              </property>
              <property name="text">
               <string>Hit</string>
              </property>
              <property name="alignment">
               <set>Qt::AlignCenter</set>
              </property>
             </widget>
            </item>
            <item>
             <widget class="QLabel" name="label_14">
              <property name="text">
               <string>Miss</string>
              </property>
              <property name="alignment">
               <set>Qt::AlignCenter</set>
              </property>
             </widget>
            </item>
            <item>
             <widget class="QLabel" name="label_17">
              <property name="text">
               <string>Killed</string>
              </property>
              <property name="alignment">
               <set>Qt::AlignCenter</set>
              </property>
             </widget>
            </item>
           </layout>
          </item>
         </layout>
        </item>
       </layout>
```

```
          </item>
          <item>
           <widget class="Field" name="field_ships" native="true"/>
          </item>
          <item>
           <widget class="Field" name="field_hits" native="true"/>
          </item>
         </layout>
        </widget>
       </widget>
       <widget class="QMenuBar" name="menubar">
        <property name="geometry">
         <rect>
          <x>0</x>
          <y>0</y>
          <width>1386</width>
          <height>21</height>
         </rect>
        </property>
       </widget>
       <widget class="QStatusBar" name="statusbar"/>
      </widget>
      <customwidgets>
       <customwidget>
        <class>Field</class>
        <extends>QWidget</extends>
        <header>field.h</header>
        <container>1</container>
       </customwidget>
      </customwidgets>
      <resources/>
      <connections/>
     </ui>
```

Сервер:

Файл «main.c»:

```
#include<header.h>

int main()
{
    NetworkContext *network_context = _network_init (PORT);

    if(!network_context){
        printf ("Failed to initialize server\n");
        return -1;
    }

    Player player1 = receive_username (network_context);

    printf ("Adding the first player with username: %s\n",
player1.username);

    int current_player = First_player;

    send_state (current_player, player1, network_context);
```

```
        Player player2 = receive_username(network_context);

        printf ("Adding the second player with username: %s\n",
player2.username);

        send_username (player2.username, network_context, player1);

        current_player = Second_player;

        send_state (current_player, player2, network_context);

        send_username (player1.username, network_context, player2);

        current_player = First_player;

        Move move;
        Answer answer;

        for(;;){
            switch (current_player){

            case First_player:

                while(1){
                    move = receive_move (network_context, player1);

                    int res;

                    if(move.line >= 0 && move.column >= 0 && move.line <
10 && move.column < 10)
                        res = 1;
                    else res = -1;

                    send_confirm(res, network_context, player1);

                    if (res > 0)
                        break;
                }

                printf ("%s: %d:%d\n", player1.username, move.line,
move.column);

                send_move (move, network_context, player2);

                answer = receive_answer (network_context, player2);
                send_answer(answer, network_context, player1);

                printf ("%s: result ", player2.username);

                switch (answer.result_of_shoot) {

                case Miss:
                    printf ("Miss\n");
                    break;

                case Hit:
                    printf ("Hit\n");
                    break;
```

```
                case Killed:
                    printf ("Killed\n");
                    break;

                default:
                    printf ("%d\n", answer.result_of_shoot);
                    break;
                }

                if (answer.result_of_shoot != Hit &&
answer.result_of_shoot != Killed){
                    printf ("\nMove goes to %s\n\n", player2.username);
                    current_player = Second_player;
                }

                if(answer.result_of_shoot == Killed &&
answer.is_any_live_ship == 0)
                    current_player = Winner_is_determined;

                break;


            case Second_player:

                while(1){
                    move = receive_move (network_context, player2);

                    int res;

                    if(move.line >= 0 && move.column >= 0 && move.line <
10 && move.column < 10)
                        res = 1;
                    else res = -1;

                    send_confirm (res, network_context, player2);

                    if (res > 0)
                        break;
                }
                printf ("%s: %d:%c\n", player2.username, move.line,
move.column + 65);

                send_move (move, network_context, player1);

                answer = receive_answer (network_context, player1);
                send_answer (answer, network_context, player2);

                printf ("%s: result ", player1.username);

                switch (answer.result_of_shoot) {

                case Miss:
                    printf ("Miss\n");
                    break;

                case Hit:
                    printf ("Hit\n");
                    break;

                case Killed:
```

```
                    printf ("Killed\n");
                    break;

                default:
                    printf ("%d\n", answer.result_of_shoot);
                    break;
                }

                if (answer.result_of_shoot != Hit &&
answer.result_of_shoot != Killed){
                    current_player = First_player;
                    printf ("\nMove goes to %s\n\n", player1.username);
                }

                if(answer.result_of_shoot == Killed &&
answer.is_any_live_ship == 0)
                    current_player = Winner_is_determined;

                break;

            }

            if(current_player == Winner_is_determined)
                break;

        }

        close (network_context->sockfd);
        free(network_context);
        return 0;
    }
```

Файл «network.h»:

```
    #ifndef NETWORK_H
    #define NETWORK_H


    #include <stdio.h>
    #include <stdlib.h>
    #include <unistd.h>
    #include <string.h>
    #include <sys/types.h>
    #include <sys/socket.h>
    #include <arpa/inet.h>
    #include <netinet/in.h>

    #define MAXLINE 1024
    #define PORT     8080

    enum CurrentPlayer
    {
        First_player = 1,
        Second_player = 2,
        Winner_is_determined = 3
    };

    enum Condition
```

```
    {
        ERROR = -1,
        Empty = 0,
        Neighbor = 1,
        Engaged = 2,
        Miss = 3,
        Hit = 4,
        Killed = 5
    };

    struct Move
    {
        int line;
        int column;
    };

    typedef struct Move Move ;

    struct Answer
    {
        int result_of_shoot;
        int is_any_live_ship;
    };

    typedef struct Answer Answer;

    struct Player
    {
        char username[MAXLINE];
        struct sockaddr_in cliaddr;
        int len;
    };

    typedef struct Player Player;

    struct NetworkContext
    {
        int sockfd;
        struct sockaddr_in servaddr;
    };

    typedef struct NetworkContext NetworkContext;

    NetworkContext *_network_init(uint16_t port);

    Player receive_username(NetworkContext *ctx);

    Move receive_move(NetworkContext *ctx, Player player);

    Answer receive_answer(NetworkContext *ctx, Player player);

    void send_state(int state, Player player, NetworkContext *ctx);

    void send_username(char username[MAXLINE], NetworkContext *ctx,
Player player);

    void send_confirm(int res, NetworkContext *ctx, Player player);

    void send_move(Move move, NetworkContext *ctx, Player player);
```

```
void send_answer(Answer answer, NetworkContext *ctx, Player player);

#endif // NETWORK_H
```

Файл «network,c»:

```
#include "network.h"
NetworkContext *_network_init(uint16_t port)
{

    NetworkContext *ctx = malloc (sizeof (NetworkContext));
    if(!ctx)
        goto error1;

    int sock = socket (AF_INET, SOCK_DGRAM, IPPROTO_UDP);

    if(sock < 0){
        perror("socket creation failed");
        goto error2;
    }

    ctx->sockfd = sock;
    memset(&ctx->servaddr, 0, sizeof(ctx->servaddr));
    ctx->servaddr.sin_family = AF_INET; // IPv4
    ctx->servaddr.sin_addr.s_addr = INADDR_ANY;
    ctx->servaddr.sin_port = htons(port);

    int bind_status = bind (ctx->sockfd,
                            (const struct sockaddr *) &ctx->servaddr,
                            sizeof(ctx->servaddr));

    if(bind_status < 0){
        perror("bind failed");
        goto error3;
    }

    return ctx;

error3:
    close (ctx->sockfd);

error2:
    free(ctx);

error1:
    return NULL;
}


Player receive_username(NetworkContext *ctx)
{
    Player player;
    memset(&player.cliaddr, 0, sizeof(player.cliaddr));
    player.len = sizeof (player.cliaddr);

    recvfrom (ctx->sockfd,
              (char *)player.username,
```

```
                MAXLINE,
                MSG_WAITALL,
                (struct sockaddr *)&player.cliaddr,
                (socklen_t *)&player.len);
        return player;
    }

    Move receive_move(NetworkContext *ctx, Player player)
    {
        Move move;

        recvfrom (ctx->sockfd,
                &move,
                sizeof (Move),
                0,
                (struct sockaddr *)&player.cliaddr,
                (socklen_t *)&player.len);

        return move;
    }

    Answer receive_answer(NetworkContext *ctx, Player player)
    {
        Answer answer;

        recvfrom (ctx->sockfd,
                &answer,
                sizeof (Answer),
                0,
                (struct sockaddr *)&player.cliaddr,
                (socklen_t *)&player.len);

        return answer;
    }

    void send_state(int state, Player player, NetworkContext *ctx)
    {
        sendto (ctx->sockfd,
                &state,
                sizeof (int),
                MSG_WAITALL,
                (const struct sockaddr *)&player.cliaddr,
                player.len);
    }

    void send_username(char username[MAXLINE], NetworkContext *ctx,
Player player)
    {
        sendto (ctx->sockfd,
                (const char *) username,
                strlen (username),
                MSG_WAITALL,
                (const struct sockaddr *)&player.cliaddr,
                player.len);
    }

    void send_confirm(int res, NetworkContext *ctx, Player player)
    {
        sendto (ctx->sockfd,
                &res,
```

```
            sizeof (int),
            0,
            (const struct sockaddr *)&player.cliaddr,
            player.len);
}

void send_move(Move move, NetworkContext *ctx, Player player)
{
    sendto (ctx->sockfd,
            &move,
            sizeof (Move),
            0,
            (const struct sockaddr *)&player.cliaddr,
            player.len);
}

void send_answer(Answer answer, NetworkContext *ctx, Player player)
{
    sendto (ctx->sockfd,
            &answer,
            sizeof (Answer),
            0,
            (const struct sockaddr *)&player.cliaddr,
            player.len);
}
```