Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 “Компьютерные науки и прикладная математика”

Кафедра №806 “Вычислительная математика и программирование”

**Курсовой проект**

**«Морской бой на memory** **map»**

Группа: М80-206Б-22

Студентка: Варнак Е. В.

Преподаватель: Миронов Е.С.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 07.01.24

Москва, 2023

**Постановка задачи**

Необходимо спроектировать и реализовать программный прототип в соответствии с выбранным вариантом. Произвести анализ и сделать вывод на основании данных, полученных при работе программного прототипа.

Проектирование консольной клиент-серверной игры

Создать собственную игру более, чем для одного пользователя. Игра может быть устроена по принципу: клиент-клиент, сервер-клиент. Необходимо написать 2 программы: сервер и клиент. Сначала запускается сервер, а далее клиенты соединяются с сервером. Сервер координирует клиентов между собой. При запуске клиента игрок может выбрать одно из следующих действий (возможно больше, если предусмотрено вариантом):

1. Создать игру, введя ее имя,
2. Присоединиться к одной из существующих игр по имени игры.

Морской бой. Общение между сервером и клиентом необходимо организовать при помощи memory map. Каждый игрок должен при запуске ввести свой логин. Должна быть предоставлена возможность отправить приглашение на игру другому игроку по логину.

**Общий метод и алгоритм решения**

Игра основана на взаимодействии клиентов и сервера, передаче сообщений между ними и их обработке. Пользователям будет предложено ввести своё имя в начале игры. Это будет “ник” игрока, который можно изменить в процессе игры, введя определённую команду. Для того чтобы вступить в игру, необходимо её создать, или присоединиться к ней, введя специальный код. Тема морского боя - пиратское сражение, инструкции к игре включены непосредственно в код проекта. Сначала запускается сервер, затем игроки запускают исполняемые файлы клиентов.

Реализация представления поля боя находится в файле “game\_functions.hpp”, там же определены функции для проверки попадания пушечного ядра во вражеский корабль и печати поля на экран.

**game\_functions.hpp**

#pragma once

#include <ctime>

#include <iostream>

#include <vector>

#include <algorithm>

#include <string>

using *namespace* std;

*bool* realkill(vector<vector<*char*>> &*field*, *int* *n*, *char* *c*){

*bool* real = true;

    if (*field*[*n*][*int*(*c*) - *int*('A')] == 'W'){

        if (*c* == 'B' || *c* == 'A' ){

            real = true;

        } else if (*field*[*n*][*int*(*c*) - *int*('A') - 1] == 'X'){

            return false;

        } else if (*field*[*n*][*int*(*c*) - *int*('A') - 1] == 'W'){

            if (*c* == 'C'){

                real = true;

            } else if (*field*[*n*][*int*(*c*) - *int*('A') - 2] == 'X'){

                return false;

            }

        }

    }

    if (*field*[*n* - 1][*int*(*c*) - *int*('A') + 1] == 'W'){

        if (*n* == 2){

            real = true;

        } else if (*field*[*n* - 2][*int*(*c*) - *int*('A') + 1] == 'X'){

            return false;

        } else if (*field*[*n* - 2][*int*(*c*) - *int*('A') + 1] == 'W'){

            if (*n* == 3){

                real = true;

            } else if (*field*[*n* - 3][*int*(*c*) - *int*('A') + 1] == 'X'){

                return false;

            }

        }

    }

    if (*field*[*n*][*int*(*c*) - *int*('A') + 2] == 'W'){

        if (*c* == 'I'){

            real = true;

        } else if (*field*[*n*][*int*(*c*) - *int*('A') + 3] == 'X'){

            return false;

        } else if (*field*[*n*][*int*(*c*) - *int*('A') + 3] == 'W'){

            if (*c* == 'H'){

                real = true;

            } else if (*field*[*n*][*int*(*c*) - *int*('A') + 4] == 'X'){

                return false;

            }

        }

    }

    if (*field*[*n* + 1][*int*(*c*) - *int*('A') + 1] == 'W'){

        if (*n* == 9){

            real = true;

        } else if (*field*[*n* + 2][*int*(*c*) - *int*('A') + 1] == 'X'){

            return false;

        } else if (*field*[*n* + 2][*int*(*c*) - *int*('A') + 1] == 'W'){

            if (*n* == 8){

                real = true;

            } else if  (*field*[*n* + 3][*int*(*c*) - *int*('A') + 1] == 'X'){

                return false;

            }

        }

    }

    return real;

}

*bool* ifkill(vector<vector<*char*>> &*field*, *int* *n*, *char* *c*){

*bool* k = false;

    if (*field*[*n*][*int*(*c*) - *int*('A') + 1] == 'X'){

        if (*field*[*n*][*int*(*c*) - *int*('A')] != 'X' &&

*field*[*n* - 1][*int*(*c*) - *int*('A')] != 'X' &&

*field*[*n* - 1][*int*(*c*) - *int*('A') + 1] != 'X' &&

*field*[*n* - 1][*int*(*c*) - *int*('A') + 2] != 'X' &&

*field*[*n*][*int*(*c*) - *int*('A') + 2] != 'X' &&

*field*[*n* + 1][*int*(*c*) - *int*('A') + 2] != 'X' &&

*field*[*n* + 1][*int*(*c*) - *int*('A') + 1] != 'X' &&

*field*[*n* + 1][*int*(*c*) - *int*('A')] != 'X'){

            if (realkill(*field*, *n*, *c*)){

                k = true;

            }

        }

    }

    return k;

}

*bool* ifhit(vector<vector<*char*>> &*field*, *int* *n*, *char* *c*){

*bool* h = false;

    if ((*field*[*n*][*int*(*c*) - *int*('A') + 1] == 'X') && (!ifkill(*field*, *n*, *c*))){

        h = true;

    }

    return h;

}

*bool* ifmiss(vector<vector<*char*>> &*field*, *int* *n*, *char* *c*){

*bool* m = false;

    if (*field*[*n*][*int*(*c*) - *int*('A') + 1] == '.'){

        m = true;

    }

    return m;

}

*bool* ifwinner(vector< vector<*char*>> &*field*) {

    for (*int* i = 1; i < 11; ++i) {

        for (*int* j = 1; j < 11; ++j) {

            if (*field*[i][j] == 'X') {

                return false;

            }

        }

    }

    return true;

}

*bool* isValidPosition(const std::vector<std::vector<*char*>>& *field*, *int* *row*, *int* *col*, *int* *shipLength*, *char* *direction*) {

    // Проверяем, попадает ли корабль в границы поля

    if (*direction* == 'h') {

        if (*col* + *shipLength* > 10) {

            return false;

        }

    } else {

        if (*row* + *shipLength* > 10) {

            return false;

        }

    }

    // Проверяем, свободны ли клетки для размещения корабля

    if (*direction* == 'h') {

        for (*int* i = 0; i < *shipLength*; ++i) {

            // Проверка клеток корабля

            if (*field*[*row*][*col* + i] != '.') {

                return false;

            }

            // Проверка соседних клеток

            for (*int* dx = -1; dx <= 1; ++dx) {

                for (*int* dy = -1; dy <= 1; ++dy) {

*int* newRow = *row* + dx;

*int* newCol = *col* + i + dy;

                    if (newRow >= 0 && newRow < 12 && newCol >= 0 && newCol < 12 && *field*[newRow][newCol] != '.') {

                        return false;

                    }

                }

            }

        }

    } else {

        for (*int* i = 0; i < *shipLength*; ++i) {

            // Проверка клеток корабля

            if (*field*[*row* + i][*col*] != '.') {

                return false;

            }

            // Проверка соседних клеток

            for (*int* dx = -1; dx <= 1; ++dx) {

                for (*int* dy = -1; dy <= 1; ++dy) {

*int* newRow = *row* + i + dx;

*int* newCol = *col* + dy;

                    if (newRow >= 0 && newRow < 12 && newCol >= 0 && newCol < 12 && *field*[newRow][newCol] != '.') {

                        return false;

                    }

                }

            }

        }

    }

    return true;

}

*void* new\_field(std::vector<std::vector<*char*>>& *field*) {

    for (*int* i = 0; i < *field*.size(); ++i) {

        for (*int* j = 0; j < *field*[i].size(); ++j) {

*field*[i][j] = '.';

        }

    }

    srand(static\_cast<*unsigned* *int*>(std::time(0))); // Инициализация генератора случайных чисел

    // Генерируем корабли случайным образом

    std::vector<*int*> shipLengths = {4, 3, 3, 2, 2, 2, 1, 1, 1, 1}; // Длины кораблей

    for (*int* i = 0; i < shipLengths.size(); ++i) {

*int* shipLength = shipLengths[i];

*int* row, col;

*char* direction;

        // Генерируем случайные координаты и направление для корабля

        do {

            row = 1 + rand() % 10;

            col = 1 + rand() % 10;

            direction = rand() % 2 == 0 ? 'h' : 'v';

        } while (!isValidPosition(*field*, row, col, shipLength, direction));

        // Размещаем корабль на поле

        if (direction == 'h') {

            for (*int* j = 0; j < shipLength; ++j) {

*field*[row][col + j] = 'X';

            }

        } else {

            for (*int* j = 0; j < shipLength; ++j) {

*field*[row + j][col] = 'X';

            }

        }

    }

}

*void* print( vector< vector<*char*>> &*field*) {

    cout << "  ";

    for (*int* i = 0; i < 10; ++i){

        cout << *char*(*int*('A')+i) << " ";

    }

    cout << endl;

    for (*int* i = 0; i < *field*.size(); ++i) {

        cout << i+1 << " ";

        for (*int* j = 0; j < *field*[i].size(); ++j) {

            std::cout << *field*[i][j] << ' ';

        }

        std::cout << std::endl;

    }

}

Весь код файла server.cpp находится в репозитории. Ниже представлена обработка сообщений с некоторой командой, пример с изменением “ника” пирата.

**server.cpp**

} else if (message\_from\_client[2] == "newname"){

            if (game.created){

                if (pirat1.name == message\_from\_client[1]){

                    pirat1.name = message\_from\_client[3];

                    string message\_to\_client = string("client:") + message\_from\_client[1] + ":changedname:" + message\_from\_client[3] + ":";

                    sprintf(mmapdata, "%s", message\_to\_client.c\_str());

                    cout << "Отправлено сообщение: " << message\_to\_client << endl;

                } else if (pirat2.name == message\_from\_client[1]){

                    pirat2.name = message\_from\_client[3];

                    string message\_to\_client = string("client:") + message\_from\_client[1] + ":changedname:" + message\_from\_client[3] + ":";

                    sprintf(mmapdata, "%s", message\_to\_client.c\_str());

                    cout << "Отправлено сообщение: " << message\_to\_client << endl;

                }

            } else {

                string message\_to\_client = string("client:") + message\_from\_client[1] + ":notchangedname:";

                sprintf(mmapdata, "%s", message\_to\_client.c\_str());

                cout << "Отправлено сообщение: " << message\_to\_client << endl;

            }

        }  else if (message\_from\_client[2] == "connection"){

            if (game.connected) {

                string message\_to\_client = string("client:") + message\_from\_client[1] + ":c\_error:";

                sprintf(mmapdata, "%s", message\_to\_client.c\_str());

                cout << "Отправлено сообщение: " << message\_to\_client << endl;

Весь код файла client.cpp находится в репозитории.

**client.cpp**

#include <fstream>

#include <mutex>

#include "game\_functions.hpp"

using *namespace* std;

//std::set<string> characters;

std::mutex mtx;

string name, password, name2, gamename = ""; // ник игрока и пароль, ник второго игрока, название игры

*bool* check\_vovel(string *c*){

*bool* fl = false;

    if (*c* == "A" || *c* == "B" || *c* == "C" || *c* == "D" || *c* == "E" || *c* == "F" || *c* == "G" || *c* == "H" || *c* == "I" || *c* == "J") {

        fl = true;

    }

    // if (c == "a" || c == "b" || c == "c" || c == "d" || c == "e" || c == "f" || c == "g" || c == "h" || c == "i" || c == "j") {

    //     fl = true;

    // }

    return fl;

}

*void* instruction() {

    cout << endl;

    cout << "Хэй, пират! Возьми вражеский корабль на абордаж!" << endl << endl;

    cout << "Инструкция для успешной игры:" << endl;

    cout << "--- create <название игры> - приготовься к сражению, начни игру" << endl;

    cout << "--- connection <название игры> - присоединение к серверу" << endl;

    cout << "--- newname <new\_name> - поменять ник, у тебя новое прозвище? Для начала вступи в бой!" << endl;

    cout << "--- invite <ник второго пирата> - карамба! что за осьминога океан принёс? вызови другого игрока на бой" << endl;

    cout << "--- checking the invitation - если тебя вызвал противник на бой, проверь, успешло ли ты присоединился к игре" << endl;

    cout << "--- <буква - горизонталь> <цифра - вертикаль> - пали изо всех пушек!" << endl;

    cout << "--- print - твой корабль затопили? Проверь, напечатай карту своего флота и карту пирата-разведчика" << endl;

    cout << "--- disconnect - отключиться от сервера" << endl;

    cout << "--- exit - выход из программы, ты сдаёшься? Не так быстро! Для начала отключись от сервера." << endl;

    cout << "--- instruction - прочитай инструкции к игре... первое правило - у настоящих пиратов нет правил!" << endl;

    cout << endl;

}

*int* mmapfd = shm\_open("game\_size", O\_RDWR, 0777); // открытие области разделяемой памяти

*char*\* mmapdata = (*char*\*)mmap(0, 10000, PROT\_READ | PROT\_WRITE, MAP\_SHARED, mmapfd, 0); // отображение области разделяемой памяти на адресное пространство процесса (возвращает указатель на начало отображ памяти)

*void* message\_send(const string &*m*) {

    mtx.lock(); // блокировка мьютекса

    memset(mmapdata, '\0', 10000); // установка \0 в разделяемую память, чтобы очистить предыдущее сообщение (если оно было)

    sprintf(mmapdata, "%s", *m*.c\_str()); // запись сообщения message в область отображенной памяти

    mtx.unlock(); // разблокировка мьютекса после записи сообщения

}

*bool* play = false;

*bool* unlocking(*int* *p*){

    if (*p* == 1){

        play = true;

    } else {

        play = false;

    }

    mtx.unlock();

    return true;

}

*bool* message\_get() { // получение ответа от сервера

    if (mmapdata[0] != 'c') {

        return false; // проверка на то, что сообщение записано для клиента

    }

    string message = mmapdata; // преобразование данных из отображенной памяти в строку

    vector<string> message\_from\_server; // вектор команд сервера

    string strings = "";

    for (*int* i = 0; i < message.size(); i++) { // по каждому символу в сообщении

        if (message[i] == ':') {

            message\_from\_server.push\_back(strings); // заполняем вектор команд

            strings = "";

        }

        else {

            strings.push\_back(message[i]); // заполняем команду

        }

    }

    if(message\_from\_server[1] == "print"){

        for (*int* i=2; i < message\_from\_server.size(); ++i){ // вывод поля (может сделать void?)

            cout << message\_from\_server[i] << endl;

        }

        return true;

    } else if (message\_from\_server[1] == name){

        mtx.lock();

        memset(mmapdata, '\0', 10000);

        if (message\_from\_server[2] == "createdgame" || message\_from\_server[2] == "connected") {

            cout << "Гром и молния! Ты начинаешь бой! Игра успешно создана." << endl;

            cout << "Посмотри на расположение своих кораблей, введи команду print." << endl;

            return unlocking(1);

        } else if (message\_from\_server[2] == "checked") {

            cout << "Пришло время сойтись якорями! Тебя вызвали на бой." << endl;

            gamename = message\_from\_server[3];

            password = message\_from\_server[4];

            return unlocking(1);

        } else if (message\_from\_server[2] == "check\_error") {

            cout << "Ты дрейфуешь в океане, бой остался за горизонтом. Подключиться к серверу не удалось." << endl;

            mtx.unlock();

            return true;

        } else if (message\_from\_server[2] == "changedname"){

            cout << "Имя успешно изменено на " << message\_from\_server[3] << endl;

            name = message\_from\_server[3];

            mtx.unlock();

            return true;

        } else if (message\_from\_server[2] == "notchangedname"){

            cout << "Имя не изменено, для доступа к имени вступи в игру." << endl;

            mtx.unlock();

            return true;

        } else if (message\_from\_server[2] == "game\_not\_connected") {

            cout << "Ждем появление вражеского корабля на горизонте, пират не борется сам с собой." << endl;

            return unlocking(1);

        } else if (message\_from\_server[2] == "game\_name\_error") {

            cout << "Проклятье медузы! Что за битва? Такой еще нет!" << endl;

            gamename = "";

            return unlocking(0);

        } else if (message\_from\_server[2] == "p\_error") {

            cout << "Тайный код неверный! Где-то бегает портовая крыса!" << endl;

            gamename = "";

            return unlocking(0);

        } else if (message\_from\_server[2] == "turn\_error") {

            cout << "Хорошая попытка, но будем мутить шторм позже, сейчас не твоя очередь!" << endl;

            return unlocking(1);

        } else if (message\_from\_server[2] == "kill") {

            cout << "УБИЛ - Ты отправил на дно вражеский корабль! Пли еще раз!" << endl;

            return unlocking(1);

        } else if (message\_from\_server[2] == "hit") {

            cout << "ПОПАЛ - Корма вражеского корабля повреждена! Пли еще раз!" << endl;

            return unlocking(1);

            cin >> password;

            if (play == false){

                gamename = game;

                string message\_to\_server = "server:" + name + ":connection:" + game + ":"+ password + ":";

                waiting\_answer(message\_to\_server);

            } else {

                cout << "Фок-грот-брамсель в левое ухо! Ты уже присоединился к серверу." << endl;

                continue;

            }

        } else if (command == "newname"){

            string newn;

            cin >> newn;

            string message\_to\_server = "server:" + name + ":newname:" + newn + ":";

            waiting\_answer(message\_to\_server);

**Протокол работы программы**

**Тестирование:**

kateland@DESKTOP-0GN4T39:~/OSLabs/FKP$ ./server

Сервер начал свою работу... Вызови пирата на бой!

mutex lock (87)

Получено пиратское сообщение: server:k:creategame:game:777:

имя первого

имя второго

Отправлено сообщение: client:k:createdgame:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:vorobey:invite:game:5:

имя первого k

имя второго

Отправлено сообщение: client:vorobey:invited:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:vorobey:checking:

имя первого k

имя второго

Отправлено сообщение: client:vorobey:checked:game:5:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:vorobey:print:

имя первого k

имя второго vorobey

Напечатаны: field1, field2

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:print:

имя первого k

имя второго vorobey

Напечатаны: field1, field2

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:B:1:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:kill:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:E:1:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:kill:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:H:2:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:hit:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:I:2:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:kill:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:E:3:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:hit:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:F:3:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:hit:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:print:

имя первого k

имя второго vorobey

Напечатаны: field1, field2

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:G:3:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:miss:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:C:3:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:turn\_error:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:vorobey:ABC:game:A:4:

имя первого k

имя второго vorobey

Отправлено сообщение: client:vorobey:miss:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:C:3:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:miss:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:vorobey:ABC:game:A:5:

имя первого k

имя второго vorobey

Отправлено сообщение: client:vorobey:miss:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:ABC:game:D:3:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:kill:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:k:disconnect:game:

имя первого k

имя второго vorobey

Отправлено сообщение: client:k:disconnected:

mutex unlock

mutex lock (87)

Получено пиратское сообщение: server:vorobey:disconnect:game:

имя первого k

имя второго vorobey

Отправлено сообщение: client:vorobey:disconnected:

mutex unlock

kateland@DESKTOP-0GN4T39:~/OSLabs/FKP$ ./client

Хэй, пират! Возьми вражеский корабль на абордаж!

Инструкция для успешной игры:

--- create <название игры> - приготовься к сражению, начни игру

--- connection <название игры> - присоединение к игре

--- newname <new\_name> - поменять ник, у тебя новое прозвище? Для начала вступи в бой!

--- invite <ник второго пирата> - карамба! что за осьминога океан принёс? вызови другого игрока на бой

--- checking the invitation - если тебя вызвал противник на бой, проверь, успешло ли ты присоединился к игре

--- <буква - горизонталь> <цифра - вертикаль> - пали изо всех пушек!

--- print - твой корабль затопили? Проверь, напечатай карту своего флота и карту пирата-разведчика

--- disconnect - отключиться от сервера

--- exit - выход из программы, ты сдаёшься? Не так быстро! Для начала отключись от сервера.

--- instruction - прочитай инструкции к игре... первое правило - у настоящих пиратов нет правил!

Тысяча чертей! Пора в бой, пират!

Как тебя будет звать противник? Введи своё имя

пират$ k

k, мы начинаем...

k$ create game

Придумай тайный код доступа к игре

k$ 777

Гром и молния! Ты начинаешь бой! Игра успешно создана.

Посмотри на расположение своих кораблей, введи команду print.

k$ create game

Еще один бой!? Сил флота недостаточно для второй игры.

k$ invite vorobey

Ты вызвал другого пирата на бой!

k$ print

A B C D E F G H I J

1 . . . . . X . . . X

2 X . . . . . . . . X

3 . . . . X X X X . .

4 . . . . . . . . . .

5 . . X . . . . . . X

6 X . . . . . . . . X

7 X . X . X . . X . .

8 X . X . X . . . . .

9 . . . . X . . . . .

10 . . . . . . . . . .

A B C D E F G H I J

1 . . . . . . . . . .

2 . . . . . . . . . .

3 . . . . . . . . . .

4 . . . . . . . . . .

5 . . . . . . . . . .

6 . . . . . . . . . .

7 . . . . . . . . . .

8 . . . . . . . . . .

9 . . . . . . . . . .

10 . . . . . . . . . .

k$ B 1

B 1

УБИЛ - Ты отправил на дно вражеский корабль! Пли еще раз!

k$ E 1

E 1

УБИЛ - Ты отправил на дно вражеский корабль! Пли еще раз!

k$ H 2

H 2

ПОПАЛ - Корма вражеского корабля повреждена! Пли еще раз!

k$ I 2

I 2

УБИЛ - Ты отправил на дно вражеский корабль! Пли еще раз!

k$ E 3

E 3

ПОПАЛ - Корма вражеского корабля повреждена! Пли еще раз!

k$ F 3

F 3

ПОПАЛ - Корма вражеского корабля повреждена! Пли еще раз!

k$ print

A B C D E F G H I J

1 . . . . . X . . . X

2 X . . . . . . . . X

3 . . . . X X X X . .

4 . . . . . . . . . .

5 . . X . . . . . . X

6 X . . . . . . . . X

7 X . X . X . . X . .

8 X . X . X . . . . .

9 . . . . X . . . . .

10 . . . . . . . . . .

A B C D E F G H I J

1 . K . . K . . . . .

2 . . . . . . . W K .

3 . . . . W W . . . .

4 . . . . . . . . . .

5 . . . . . . . . . .

6 . . . . . . . . . .

7 . . . . . . . . . .

8 . . . . . . . . . .

9 . . . . . . . . . .

10 . . . . . . . . . .

k$ G 3

G 3

МИМО - Салют для вражеских кораблей! Пустая трата пушечных ядер, ты промахнулся!

k$ C 3

C 3

Хорошая попытка, но будем мутить шторм позже, сейчас не твоя очередь!

k$ C 3

C 3

МИМО - Салют для вражеских кораблей! Пустая трата пушечных ядер, ты промахнулся!

k$ D 3

D 3

УБИЛ - Ты отправил на дно вражеский корабль! Пли еще раз!

k$ disconnect

Ты отключился от сервера.

k$ exit

kateland@DESKTOP-0GN4T39:~/OSLabs/FKP$ ./client

Хэй, пират! Возьми вражеский корабль на абордаж!

Инструкция для успешной игры:

--- create <название игры> - приготовься к сражению, начни игру

--- connection <название игры> - присоединение к серверу

--- newname <new\_name> - поменять ник, у тебя новое прозвище? Для начала вступи в бой!

--- invite <ник второго пирата> - карамба! что за осьминога океан принёс? вызови другого игрока на бой

--- checking the invitation - если тебя вызвал противник на бой, проверь, успешло ли ты присоединился к игре

--- <буква - горизонталь> <цифра - вертикаль> - пали изо всех пушек!

--- print - твой корабль затопили? Проверь, напечатай карту своего флота и карту пирата-разведчика

--- disconnect - отключиться от сервера

--- exit - выход из программы, ты сдаёшься? Не так быстро! Для начала отключись от сервера.

--- instruction - прочитай инструкции к игре... первое правило - у настоящих пиратов нет правил!

Тысяча чертей! Пора в бой, пират!

Как тебя будет звать противник? Введи своё имя

пират$ vorobey

vorobey, мы начинаем...

vorobey$ checking the invitation

Пришло время сойтись якорями! Тебя вызвали на бой.

vorobey$ print

A B C D E F G H I J

1 . X . . X . . . . .

2 . . . . . . . X X .

3 . X . X X X . . . .

4 . . . . . . . X X .

5 . . . . . . . . . .

6 . . X X X . . . . X

7 . . . . . . . . . .

8 . . . . . . . . . .

9 . X X . X X X X . .

10 . . . . . . . . . .

A B C D E F G H I J

1 . . . . . . . . . .

2 . . . . . . . . . .

3 . . . . . . . . . .

4 . . . . . . . . . .

5 . . . . . . . . . .

6 . . . . . . . . . .

7 . . . . . . . . . .

8 . . . . . . . . . .

9 . . . . . . . . . .

10 . . . . . . . . . .

vorobey$ A 4

A 4

МИМО - Салют для вражеских кораблей! Пустая трата пушечных ядер, ты промахнулся!

vorobey$ A 5

A 5

МИМО - Салют для вражеских кораблей! Пустая трата пушечных ядер, ты промахнулся!

vorobey$ disconnect

Ты отключился от сервера.

vorobey$ exit

**Вывод**

В ходе выполнения курсового проекта я научилась реализовывать клиент-серверное взаимодействие, создавать поле для морского боя и писать различные функции для игры. Теперь я знаю наизусть все правила “морского боя”, поскольку сыграть в эту игру мне пришлось очень много раз. Сейчас я имею больше представления о том, как создаются игры и почему могут возникнуть трудности в разработке. Эта сфера мне очень нравится, поэтому я планирую в будущем погрузиться в нее. Курс операционных систем показал, что за сложным взаимодействием клиентов и серверов, скрывается множество простых процессов и операций, на которые разбивается код. Со временем начинаешь понимать, что высокие уровни абстракции в языках программирования, это хорошо, но чем проще операции, тем проще составить картину взаимодействия и понять, как следует обработать тот или иной запрос.