Московский Авиационный Институт

(Национальный Исследовательский институт)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа по курсу**

**«Операционные системы»**

**«Консоль-серверная игра»**

Группа: М8О-201Б-20  
Студент: Капалин Д.С.  
Преподаватель: Зацепин Н.А.  
Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2022**Содержание**

1. Постановка задачи
2. Сведения о программе
3. Исходный код
4. Тестирование
5. Заключение

**Постановка задачи**

«Быки и коровы» (угадывать необходимо числа). Общение между сервером и клиентом необходимо организовать при помощи очередей сообщений (например, ZeroMQ). При создании каждой игры необходимо указывать количество игроков, которые будут участвовать. То есть угадывать могут несколько игроков. Должна быть реализована функция поиска игры, то есть игрок пытается войти в игру не по имени, а просто просит сервер найти ему игру.

**Сведения о программе**

Игра реализована в виде двух программ: управляющего сервера и клиента.   
Управляющие сервер хранит в себе объект класса, описывающего множество действительных игр(то есть тех, внутри которых либо происходит сам процесс игры, либо идет поиск игроков). Основное взаимодействие сервера с клиентом реализовано по модели REQ-REP. Клиент отправляет какой-то запрос серверу, например на создание новое игры, а сервер выполняет этот запрос и отправляет обратно ответ об успешности или неудаче при выполнении запроса. Однако, в сервер интегрирован и другой процесс взаимодействия: PUB-SUB. Он полезен для того, чтоб при случаях с большим числом игроков, время ожидания ответа на запрос возрастало минимально. И вместо того, чтоб, допустим, в игре на 10 человек каждый из игроков отправлял запрос о том, подключились ли все игроки сервер отправляет рассылку на всех игроков в виде структурированной последовательности сообщений, первым из которых идет сообщений со входным фильтром. Просматривая его, программа клиента понимает предназначается ли это сообщение ей или клиентам другой игры. Также в программе сервера задана функция генерации случайного 4-х значного числа для игры в “Быки и коровы”.  
Программа клиента имеет основной целью “регистрацию” нового игрока, а также отправку запросов о создании новой игры, подключению к уже существующей игре по ее имени, а также к подключению к игре без имени(сервер сам подбирает свободную).   
Кроме того, для уменьшения нагрузки на сервер сам процесс игры происходить в клиенте.   
При совпадении числа с тем, что сервер отправил игре клиент отправляет запрос на сервер о победе. Реализованы и элементы защиты. Например, на приемный сокет установлен флаг, который делает сокет недействительным спустя заданное время, тем самым исключая возможность бесконечного ожидания поиска игры или подключения других игроков. В клиенте реализован отдельный поток, нацеленный на то, чтоб “слушать” сервер и при победе кого-то другого принимать сообщения с сервера и завершать игру с выводом сообщения о проигрыше.   
Все клиент-серверные взаимодействия обеспечиваются по средствам API ZMQ.

**Исходный код**

**Сервер:**#include <iostream>

#include "zmq.hpp"

#include <sys/types.h>

#include <unistd.h>

#include <vector>

#include <string>

#include <algorithm>

#include <ctime>

struct Player {

int player\_pid;

std::string player\_name;

};

class Game {

public:

std::string game\_id;

std::vector<Player>players;

int players\_quantity;

std::string number;

int current\_players;

int cond;

Game()

{

game\_id;

players;

players\_quantity = 0;

current\_players = 0;

cond = 0;

std::string number = "";

}

void generate\_number() {

srand(static\_cast<unsigned int>(time(0)));

int a = (rand() % 10);

int b = (rand() % 10);

while (a == b) {

b = (rand() % 10);

}

int c = (rand() % 10);

while (a == c || b == c) {

c = (rand() % 10);

}

int d = (rand() % 10);

while (a == d || b == d || c == d) {

d = (rand() % 10);

}

number = std::to\_string(a) + std::to\_string(b) + std::to\_string(c) + std::to\_string(d);

}

};

std::string getName(std::string& str) {

int n = 2;

std::string name;

while (str[n] != ' ') {

name += str[n];

n += 1;

}

return name;

}

int getPid(std::string& str,int n) {

std::string pid;

for (int i = n; i < str.length(); i++) {

if (str[i] == ' ')

break;

pid += str[i];

}

int user\_pid = stoi(pid);

return user\_pid;

}

int getNum(std::string& str) {

std::string num;

int i = str.length();

while (str[i] != ' ') {

num += str[i];

--i;

}

(std::reverse(num.begin(), num.end()));

int res = stoi(num);

return res;

}

std::string getPlayer(std::string& str,int n) {

std::string player\_name;

while (str[n] != ' ') {

player\_name += str[n];

n += 1;

}

return player\_name;

}

bool findgame(std::vector<Game>& g, Player player, std::string name) {

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

if (g[i].game\_id == name) {

if (g[i].current\_players + 1 <= g[i].players\_quantity) {

g[i].players.push\_back(player);

g[i].current\_players += 1;

return true;

}

}

}

return false;

}

bool checkgame(std::vector<Game>& g, std::string name) {

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

if (g[i].game\_id == name) {

if (g[i].current\_players == g[i].players\_quantity) {

return true;

}

}

}

return false;

}

bool checkname(std::string& gamename, std::vector<Game>& g) {

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

if (g[i].game\_id == gamename) {

return false;

}

}

return true;

}

void deletegame(std::string& gamename, std::vector<Game>& g) {

int iter\_position = 0;

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

if (g[i].game\_id == gamename) {

iter\_position = i;

break;

}

}

std::vector<Game>::iterator my\_iterator = g.begin();

my\_iterator += iter\_position;

g.erase(my\_iterator);

}

void sending(zmq::socket\_t &sock,std::string rep,int code) {

if (code == 0) {

zmq::message\_t reply\_topic("0");

zmq::message\_t reply(rep);

sock.send(reply\_topic, zmq::send\_flags::sndmore);

sock.send(reply, zmq::send\_flags::none);

}

if (code != 0) {

zmq::message\_t reply\_topic(std::to\_string(code));

zmq::message\_t reply(rep);

sock.send(reply\_topic, zmq::send\_flags::sndmore);

sock.send(reply, zmq::send\_flags::none);

}

}

std::string getWinner(std::vector<Game>& g, int& win\_pid) {

std::string winner;

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

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

if (g[i].players[j].player\_pid == win\_pid) {

winner = g[i].players[j].player\_name;

}

}

}

return winner;

}

std::string getCurrentstring(std::string& name, std::vector<Game>& g) {

std::string current\_players;

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

if (g[i].game\_id == name) {

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

current\_players += g[i].players[j].player\_name + " ";

}

}

}

return current\_players;

}

std::string getCurrentnumber(std::string& name, std::vector<Game>& g) {

std::string current\_players;

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

if (g[i].game\_id == name) {

current\_players = std::to\_string(g[i].current\_players) + "/" + std::to\_string(g[i].players\_quantity);

}

}

return current\_players;

}

int check\_game\_condition(std::vector<Game>& g, std::string& name) {

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

if (g[i].game\_id == name) {

if (g[i].cond == 0) {

g[i].cond = 1;

return 0;

}

}

}

return 1;

}

int main() {

zmq::context\_t ctx;

zmq::socket\_t sock\_rep(ctx,zmq::socket\_type::rep);

sock\_rep.bind("tcp://127.0.0.1:5555");

zmq::socket\_t sock\_pub(ctx, zmq::socket\_type::pub);

sock\_pub.bind("tcp://127.0.0.1:5545");

std::vector<Game>games;

while (1) {

zmq::message\_t m\_in;

sock\_rep.recv(m\_in);

std::string message = m\_in.to\_string();

std::cout << message;

if (message[0] == '1') {

int user\_pid;

std::string name;

int n = 2;

name = getName(message);

n += name.length() + 1;

std::string player\_name;

user\_pid = getPid(message, n);

n += std::to\_string(user\_pid).length()+1;

player\_name = getPlayer(message, n);

int num\_player;

num\_player = getNum(message);

if (checkname(name, games) == true) {

Game game;

Player player;

game.game\_id = name;

player.player\_name = player\_name;

player.player\_pid = user\_pid;

game.players.push\_back(player);

game.players\_quantity = num\_player;

game.current\_players += 1;

game.cond = 0;

game.generate\_number();

std::string gamenumber = game.number;

games.push\_back(game);

std::string reply = "Game: " + name + " created!";

sending(sock\_rep, reply,0);

if (game.players\_quantity == game.current\_players) {

std::string str = "ready";

zmq::message\_t topic(name);

zmq::message\_t payload(str);

zmq::message\_t game\_number(gamenumber);

sock\_pub.send(topic, zmq::send\_flags::sndmore);

sock\_pub.send(payload, zmq::send\_flags::sndmore);

sock\_pub.send(game\_number, zmq::send\_flags::none);

}

}

else {

std::string reply = "Game already exists. Please choose other name.";

sending(sock\_rep, reply,1);

}

}

if (message[0] == '2') {

int user\_pid;

std::string name;

name = getName(message);

std::string player\_name;

player\_name = getPlayer(message, 2 + name.length() + 1);

user\_pid = getPid(message, 2 + name.length() + 1+player\_name.length()+1);

Player player;

player.player\_name = player\_name;

player.player\_pid = user\_pid;

if (findgame(games, player, name) == true) {

std::string reply;

std::string current\_players = getCurrentstring(name, games);

std::string current\_number = getCurrentnumber(name, games);

reply = "Connected to game:" + name + " in lobby now: " + current\_players + current\_number;

sending(sock\_rep,reply,0);

zmq::message\_t top1(name);

zmq::message\_t pay1(player\_name);

sock\_pub.send(top1, zmq::send\_flags::sndmore);

sock\_pub.send(pay1, zmq::send\_flags::none);

if (checkgame(games, name) == true) {

std::string str = "ready";

zmq::message\_t topic(name);

zmq::message\_t payload(str);

std::string gamenumber;

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

if (games[i].game\_id == name) {

gamenumber = games[i].number;

}

}

zmq::message\_t game\_number(gamenumber);

sock\_pub.send(topic, zmq::send\_flags::sndmore);

sock\_pub.send(payload, zmq::send\_flags::sndmore);

sock\_pub.send(game\_number, zmq::send\_flags::none);

}

}

else {

std::string reply;

reply = "Game not found :(";

sending(sock\_rep, reply,1);

}

}

if (message[0] == '3') {

std::string player\_name = getPlayer(message, 2);

int user\_pid = getPid(message, 2+player\_name.length()+1);

Player player;

player.player\_name = player\_name;

player.player\_pid = user\_pid;

std::string gamename;

std::string gamenumber;

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

if (games[i].current\_players < games[i].players\_quantity) {

gamename = games[i].game\_id;

gamenumber = games[i].number;

break;

}

}

if (findgame(games, player, gamename) == true) {

std::string reply;

std::string current\_players = getCurrentstring(gamename, games);

std::string current\_number = getCurrentnumber(gamename, games);

reply = "Connected to game: " + gamename+" "+current\_players + current\_number;;

sending(sock\_rep, reply,0);

sleep(3);

zmq::message\_t top1(gamename);

zmq::message\_t pay1(player\_name);

sock\_pub.send(top1, zmq::send\_flags::sndmore);

sock\_pub.send(pay1, zmq::send\_flags::none);

if (checkgame(games, gamename) == true) {

std::string str = "ready";

zmq::message\_t topic(gamename);

zmq::message\_t payload(str);

sleep(3);

zmq::message\_t game\_number(gamenumber);

sock\_pub.send(topic, zmq::send\_flags::sndmore);

sock\_pub.send(payload, zmq::send\_flags::sndmore);

sock\_pub.send(game\_number, zmq::send\_flags::none);

}

}

else {

std::string reply;

reply = "Game not found :(";

sending(sock\_rep, reply,1);

}

}

if (message[0] == '4') {

std::string gamename = getName(message);

int win\_pid = getPid(message, gamename.length() + 3);

std::string reply = "ok";

sending(sock\_rep, reply, 0);

if (check\_game\_condition(games,gamename)==0) {

zmq::message\_t topic(gamename);

std::string pid = std::to\_string(win\_pid);

zmq::message\_t payload(pid);

std::string winner = getWinner(games, win\_pid);

zmq::message\_t payload2(winner);

sock\_pub.send(topic, zmq::send\_flags::sndmore);

sock\_pub.send(payload, zmq::send\_flags::sndmore);

sock\_pub.send(payload2, zmq::send\_flags::none);

deletegame(gamename, games);

}

}

}

return 0;

}

**Клиент:**

#include <iostream>

#include "zmq.hpp"

#include <sys/types.h>

#include <unistd.h>

#include <string>

#include <algorithm>

#include <thread>

#include <chrono>

int flag = 0;

std::string winner;

std::vector<std::string> waiter(int mode,int& current\_players, int& all\_players, zmq::socket\_t& sock\_sub, zmq::message\_t& topic, zmq::message\_t& payload,zmq::message\_t& game\_number ) {

if (payload.to\_string().length() != 0 || mode == 1) {

for (int i = current\_players; i < all\_players; i++) {

sock\_sub.recv(topic);

sock\_sub.recv(payload);

if (payload.to\_string().length() == 0) {

break;

}

else {

current\_players += 1;

std::cout << "Connected: " << payload.to\_string() << "! " << current\_players << "/" << all\_players << "\n";

}

}

int timeout = 180000;

sock\_sub.setsockopt(ZMQ\_RCVTIMEO, &timeout, sizeof(timeout));

sock\_sub.recv(topic);

sock\_sub.recv(payload);

sock\_sub.recv(game\_number);

std::cout << "game number(test)"<< game\_number.to\_string() << "\n";

std::string gamenumber = game\_number.to\_string();

std::vector<std::string> v{ payload.to\_string(),gamenumber };

return v;

}

else {

std::string s = "timeout";

std::vector<std::string> v{s};

return v;

}

}

std::string getGameName(std::string& str) {

std::string st = "Connected to game: ";

int n = st.length();

std::string num;

int i = n;

while (str[i] != ' ') {

num += str[i];

++i;

}

return num;

}

void waiting(zmq::socket\_t& sub,std::string &gamename, int& realpid) {

sub.setsockopt(ZMQ\_SUBSCRIBE, gamename.c\_str(), gamename.size());

zmq::message\_t topic;

zmq::message\_t payload;

zmq::message\_t winner\_name;

sub.recv(topic);

sub.recv(payload);

sub.recv(winner\_name);

winner = winner\_name.to\_string();

if (payload.to\_string() != std::to\_string(realpid)) {

flag += 1;

}

else if (payload.to\_string() == std::to\_string(realpid)) {

flag = -1;

}

}

int checknumber(std::string& inp,std::string &gamenumber) {

int cows = 0;

int bulls = 0;

for (int i = 0; i < 4; i++) {

for (int j = 0; j < 4; j++) {

if (inp[j] == gamenumber[i] && j == i)

bulls += 1;

if (inp[j] == gamenumber[i] && j != i) {

cows += 1;

}

}

}

if (bulls == 4) {

return 1;

}

else {

std::cout << "bulls: " << bulls << " " << "cows: " << cows << "\n";

return -1;

}

}

int game(std::string& gamenumber, zmq::socket\_t& sub, zmq::socket\_t& req,std::string& gamename) {

int realpid = getpid();

std::thread sender\_thread(waiting,std::ref(sub),std::ref(gamename),std::ref(realpid));

int win\_condition = 0;

while (flag != 1) {

std::cout << "input number: ";

std::string inp;

std::cin >> inp;

if (checknumber(inp, gamenumber) == 1) {

win\_condition += 1;

break;

}

}

if (win\_condition == 1) {

std::string message = "4 " + gamename + " "+ std::to\_string(realpid);

zmq::message\_t m\_out(message);

req.send(m\_out, zmq::send\_flags::none);

zmq::message\_t m\_in\_topic;

zmq::message\_t m\_in\_payload;

req.recv(m\_in\_topic);

req.recv(m\_in\_payload);

}

std::this\_thread::sleep\_for(std::chrono::milliseconds(1000));

if (flag == -1) {

std::cout << "Your win\n";

}

else {

std::cout << "You lose\n";

std::cout << "Winner is: " << winner << "\n";

}

sender\_thread.join();

flag = 0;

return 0;

}

int main() {

zmq::context\_t ctx;

zmq::socket\_t sock0(ctx, zmq::socket\_type::req);

sock0.connect("tcp://127.0.0.1:5555");

zmq::socket\_t sock\_sub(ctx, zmq::socket\_type::sub);

sock\_sub.connect("tcp://127.0.0.1:5545");

std::string player\_name;

std::cout << "Input your player name\n";

std::cin >> player\_name;

std::cout << "Your name is: " << player\_name << "\n";

while (1) {

std::cout << "1.Create game 2.Connect to the game 3.Find game \n";

int c;

std::cin >> c;

if (c == 1) {

std::cout << "Input game name\n"; //ввод названия игры

std::string gamename;

std::cin >> gamename;

sock\_sub.setsockopt(ZMQ\_SUBSCRIBE, gamename.c\_str(), gamename.size()); // устанрвка фильтра сообщений соекта PUB-SUB

zmq::message\_t topic; // инициализация сообщений

zmq::message\_t payload;

zmq::message\_t game\_number;

zmq::message\_t m\_in\_topic;

zmq::message\_t m\_in\_payload;

std::string number\_of\_players; // ввод кол-ва игроков

std::cout << "cin number of players\n";

std::cin >> number\_of\_players;

int my\_pid = getpid(); // получаем pid (будет нужен как id игрока у сервера)

//std::cout << "pid before sending: " << my\_pid << "\n";

std::string message = "1 " + gamename + " " + std::to\_string(my\_pid) + " " + player\_name + " " + number\_of\_players; // формируем сообщение для отправки на сервер

zmq::message\_t m\_out(message);

sock0.send(m\_out, zmq::send\_flags::none);

sock0.recv(m\_in\_topic); //принимаем ответ

sock0.recv(m\_in\_payload);

int code = 0;

int topico = stoi(m\_in\_topic.to\_string());

if (topico == code) {

std::cout << "waiting for other people in lobby...\n";

int timeout = 180000;

sock\_sub.setsockopt(ZMQ\_RCVTIMEO, &timeout, sizeof(timeout));

int h = stoi(number\_of\_players);

//std::cout << "h " << h << "\n";

int current\_players = 1;

std::vector<std::string> v = waiter(1,current\_players, h, sock\_sub, topic, payload, game\_number);

if (v[0] == "ready"){

std::cout << "the game is ready\n";

std::cout << "Game starting\n";

game(v[1], sock\_sub, sock0, gamename);

}

else {

std::cout << "error timeout\n";

std::cout << v[0];

}

}

else {

std::cout << m\_in\_payload.to\_string() << "\n";

}

}

if (c == 2) {

std::cout << "cin name of the game\n";

std::string gamename;

std::cin >> gamename;

sock\_sub.setsockopt(ZMQ\_SUBSCRIBE, gamename.c\_str(), gamename.size());

zmq::message\_t topic;

zmq::message\_t payload;

zmq::message\_t game\_number;

zmq::message\_t m\_in\_topic;

zmq::message\_t m\_in\_payload;

int my\_pid = getpid();

std::string message = "2 " + gamename + " " + player\_name+ " " + std::to\_string(my\_pid);

zmq::message\_t m\_out(message);

sock0.send(m\_out, zmq::send\_flags::none);

sock0.recv(m\_in\_topic);

sock0.recv(m\_in\_payload);

int code = 0;

int topico = stoi(m\_in\_topic.to\_string());

if (topico == code) {

std::cout << "waiting for other people\n";

std::string new\_message = m\_in\_payload.to\_string();

char str1 = new\_message[new\_message.length() - 3];

char str2 = new\_message[new\_message.length() - 1];

int current\_players = str1 - '0';

int all\_players = str2 - '0';

int timeout = 180000;

sock\_sub.setsockopt(ZMQ\_RCVTIMEO, &timeout, sizeof(timeout));

sock\_sub.recv(topic);

sock\_sub.recv(payload);

std::vector<std::string> v = waiter(0,current\_players, all\_players, sock\_sub, topic, payload, game\_number);

if (v[0] == "ready") {

std::cout << "the game is ready\n";

std::string gamenumber = game\_number.to\_string();

game(v[1], sock\_sub, sock0, gamename);

}

else {

std::cout << "error timeout\n";

}

}

else {

std::cout << m\_in\_payload << "\n";

}

}

if (c == 3) {

std::cout << "Waitging for the game\n";

int my\_pid = getpid();

std::string message = "3 " + player\_name + " " + std::to\_string(my\_pid);

zmq::message\_t m\_out(message);

sock0.send(m\_out, zmq::send\_flags::none);

zmq::message\_t m\_in\_topic;

zmq::message\_t m\_in\_payload;

sock0.recv(m\_in\_topic);

sock0.recv(m\_in\_payload);

int code = 0;

int topico = stoi(m\_in\_topic.to\_string());

if (topico == code) {

std::cout << m\_in\_payload.to\_string() << "\n";

std::cout << "waiting for other people\n";

std::string message\_in = m\_in\_payload.to\_string();

std::string gamename = getGameName(message\_in);

std::cout << gamename;

sock\_sub.setsockopt(ZMQ\_SUBSCRIBE, gamename.c\_str(), gamename.size());

zmq::message\_t topic;

zmq::message\_t payload;

zmq::message\_t game\_number;

std::string new\_message = m\_in\_payload.to\_string();

char str1 = new\_message[new\_message.length() - 3];

char str2 = new\_message[new\_message.length() - 1];

int current\_players = str1 - '0';

int all\_players = str2 - '0';

int timeout = 180000;

sock\_sub.setsockopt(ZMQ\_RCVTIMEO, &timeout, sizeof(timeout));

sock\_sub.recv(topic);

sock\_sub.recv(payload);

std::vector<std::string> v = waiter(0, current\_players, all\_players, sock\_sub, topic, payload, game\_number);

if (v[0] == "ready") {

std::cout << "the game is ready\n";

game(v[1], sock\_sub, sock0, gamename);

}

else {

std::cout << "error\n";

}

}

else {

std::cout << m\_in\_payload.to\_string() << "\n";

}

}

if (c == 4) {

exit(1);

}

}

return 0;

}

**Тестирование**

1). Запустим три клиента. Один создаст игру, другой подключиться по имени игры, третий подключиться через поиск игры.

Main

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ g++ -std=c++11 -pthread -fsanitize=address main.cpp -lzmq -o main -w

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ ./main

Input your player name

rem

Your name is: rem

1.Create game 2.Connect to the game 3.Find game

1

Input game name

das

cin number of players

3

waiting for other people in lobby...

Connected: asd! 2/3

Connected: wer! 3/3

the game is ready

Game starting

input number: 2534

bulls: 2 cows: 1

input number: 2436

Your win

1.Create game 2.Connect to the game 3.Find game

Main2

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ g++ -std=c++11 -pthread -fsanitize=address main.cpp -lzmq -o main2 -w

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ ./main2

Input your player name

asd

Your name is: asd

1.Create game 2.Connect to the game 3.Find game

2

cin name of the game

das

waiting for other people

Connected: wer! 3/3

the game is ready

input number: 2345

bulls: 1 cows: 2

input number: 2346

bulls: 2 cows: 2

input number: 2436

You lose

Winner is: rem

1.Create game 2.Connect to the game 3.Find game

Main3

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ g++ -std=c++11 -pthread -fsanitize=address main.cpp -lzmq -o main3 -w

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ ./main3

Input your player name

wer

Your name is: wer

1.Create game 2.Connect to the game 3.Find game

3

Waitging for the game

Connected to game: das rem asd wer 3/3

waiting for other people

the game is ready

input number: 2413

bulls: 2 cows: 1

input number: 2145

bulls: 1 cows: 1

You lose

Winner is: rem

1.Create game 2.Connect to the game 3.Find game

server

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ g++ -fsanitize=address server.cpp -lzmq -o server -w

aster1@LAPTOP-O390ICKU:/mnt/c/Users/K-PC/source/repos/OSKP$ ./server

2). Запустим двух клиентов. Отработаем некорректный ввод.

Main

1.Create game 2.Connect to the game 3.Find game

1

Input game name

afr

cin number of players

2

waiting for other people in lobby...

Connected: asd! 2/2

the game is ready

Game starting

input number: 1783

bulls: 3 cows: 0

You lose

Winner is: asd

1.Create game 2.Connect to the game 3.Find game

Main2

1.Create game 2.Connect to the game 3.Find game

2

cin name of the game

red

zmq::message\_t [size 017] ( Game not found :()

1.Create game 2.Connect to the game 3.Find game

3

Waitging for the game

Connected to game: afr rem asd 2/2

waiting for other people

the game is ready

input number: 1782

Your win

1.Create game 2.Connect to the game 3.Find game

1782

1.Create game 2.Connect to the game 3.Find game

3) Запустим две игры параллельно.   
Main1

1.Create game 2.Connect to the game 3.Find game

1

Input game name

wert

cin number of players

1

waiting for other people in lobby...

the game is ready

Game starting

input number: 2134

bulls: 0 cows: 2

input number: 3710

Your win

Main2

1.Create game 2.Connect to the game 3.Find game

1

Input game name

tr

cin number of players

2

waiting for other people in lobby...

Connected: wer! 2/2

the game is ready

Game starting

input number: 5671

bulls: 2 cows: 1

input number: 5687

Your win

1.Create game 2.Connect to the game 3.Find game

Main3

1.Create game 2.Connect to the game 3.Find game

2

cin name of the game

werd

zmq::message\_t [size 017] ( Game not found :()

1.Create game 2.Connect to the game 3.Find game

2

cin name of the game

wert

zmq::message\_t [size 017] ( Game not found :()

1.Create game 2.Connect to the game 3.Find game

3

Waitging for the game

Connected to game: tr asd wer 2/2

waiting for other people

the game is ready

input number: 5615

bulls: 2 cows: 1

input number: 6672

bulls: 1 cows: 2

You lose

Winner is: rem

1.Create game 2.Connect to the game 3.Find game

**Заключение**

В ходе выполнения задания курсового проекта я получил большой опыт по использованию API ZeroMQ, налаживанию сетевого взаимодействия между сервером и программами клиентами с применением различных паттернов взаимодействия( REQ-REP, PUB-SUB). Кроме того, во время реализации пришлось задуматься о различных путях снижения нагрузки на сервер и уменьшения времени ожидания ответа от сервера. Для оптимизации этих процессов потребовалось вспомнить знания, полученные в ходе выполнения ЛР 3 и произвести интеграцию в программу игрока дополнительный поток. Исходя из этого, курсовой проект был крайне интересен для реализации и привнес большое количество прикладных навыков.