Министерство науки и высшего образования 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 *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ

по дисциплине: «Разработка программных систем»

Студент	Харитонов Евгений Юрьевич
Группа	РК6-62б
Тип задания	лабораторная работа
Тема лабораторной работы	сетевое программирование

Студент _____ **Харитонов Е.Ю.**
подпись, дата *фамилия, и.о.*

Преподаватель _____ **Федорук В.Г.**
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2020 г.

Оглавление

Задание на лабораторную работу

Вариант 5:

Разработать клиент-серверное приложение "Морской бой". Играют двое пользователей-людей, обмениваясь стандартными фразами: "a7", "мимо", "ранил", "убил". Подобный протокол взаимодействия (но на формальном уровне) должно реализовывать сетевое приложение в режиме клиент-сервер (серверная сторона ожидает первого хода в ее сторону).

Описание используемого (или разработанного самостоятельно) прикладного протокола сетевого взаимодействия

Для взаимодействия между игроками используется socket-интерфейс. Один из игроков выступает сервером, второй - клиентом. Игрок, выступающий сервером, создает игру и ожидает подключения соперника. Соперник-клиент подключается к серверу по определенному IP-адресу и порту и игра начинается.

Для ходов приложения обмениваются сообщениями длиной 3 байта (3 символа), в которых одним символом закодирован результат предыдущего выстрела и 2 символами координаты для нового выстрела (поле 10 кодируется как 0). Результат предыдущего выстрела кодируется одним из символов: M - мимо, R - ранил, K - убил, S - проигрыш (отправляющий сообщение с символом S проиграл). В случае, если какое-то действие не производится, лишние поля заполняются символом N

(например, если мы ранили соперника, он не стреляет в нашу сторону, а только отвечает RNN).

Сначала соперник-клиент отправляет сообщение, далее обмен сообщениями происходит по очереди. При получении сообщения, обозначающего проигрыш соперника, соединение разрывается и игра заканчивается.

Описание структуры программы

При запуске программы предлагается выбрать: создать новую игру или подключиться к сопернику.

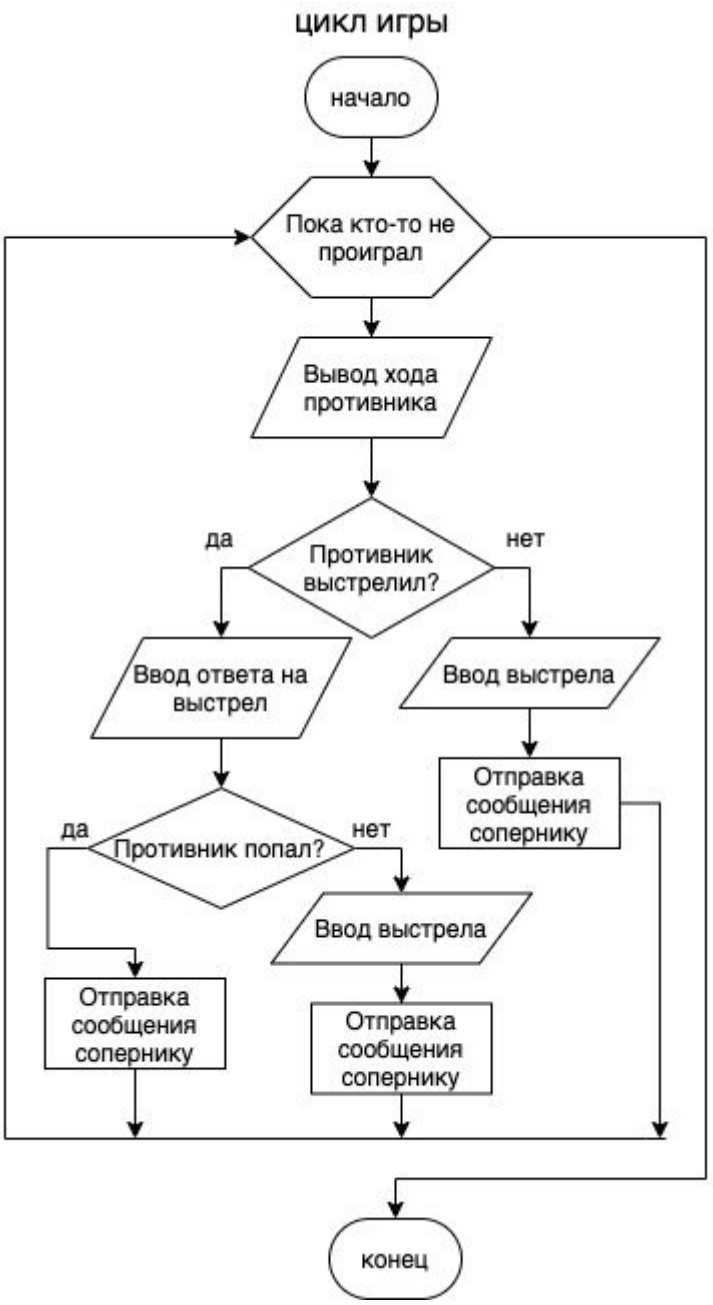
При создании новой игры создается сокет, слушающий все сетевые интерфейсы и начинается его прослушка в ожидании входящего соединения.

При подключении к сопернику мы создаем сокет и подключаемся к сопернику-серверу по IP адресу и порту, которые вводятся из стандартного потока ввода.

После установления соединения программы попадают в цикл игры. Дается возможность хода игроку-клиенту, в это время игрок-сервер ожидает, когда соперник походит. После ввода координат для выстрела в формате a1 сообщение отправляется сопернику, а игрок-клиент начинает ожидать ответного хода. Если попал, игрок-сервер отвечает “убил” или “ранен” и начинает ожидать ответ, иначе нужно ответить “мимо” и ввести координаты для выстрела

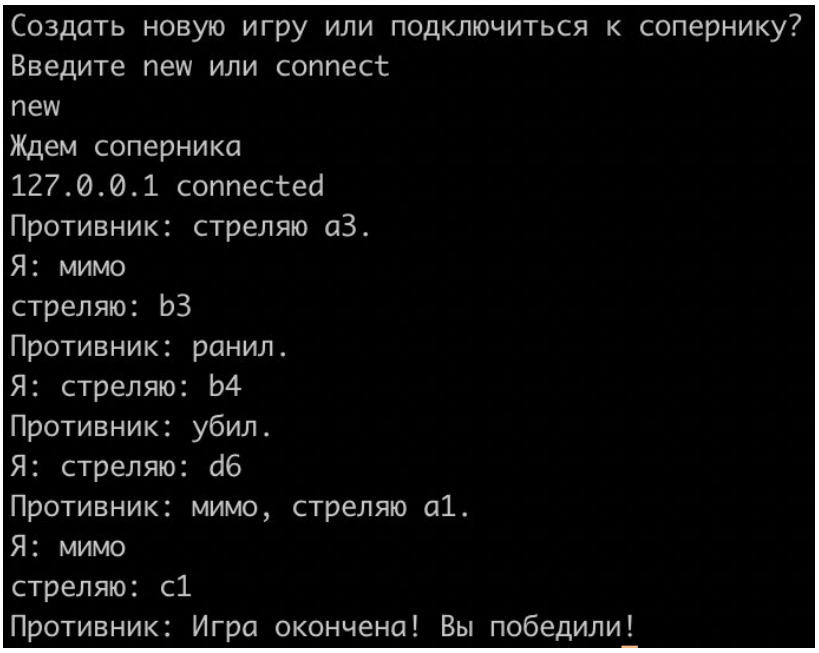
Игра продолжается, пока не будут уничтожены все корабли. Если последний корабль потоплен, нужно написать в ответ на выстрел stop, в таком случае игра закончится, соединение разорвется, а соперник узнает, что он выиграл.

Блок схема программы

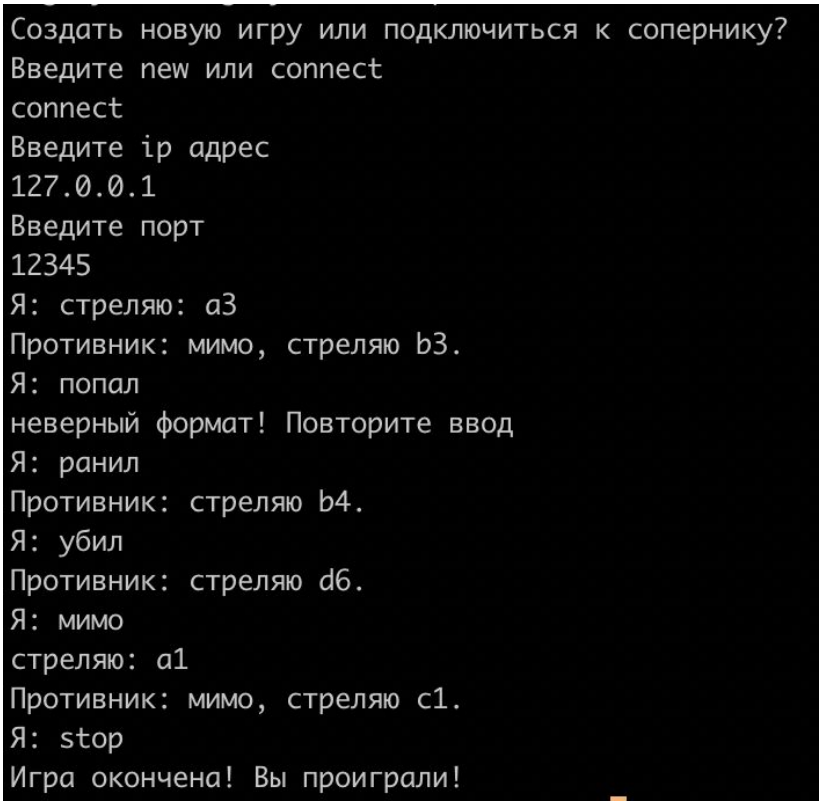


Примеры результатов работы программы

Пример игры со стороны сервера:



Пример игры со стороны клиента



Текст программы

main.c

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

#define SRV_PORT 12345
#define BUF_LEN 4
#define READ_BUF_LEN 16

void flush_stdin() {
    char c;
    while ((c = getchar()) != '\n' && c != EOF) { }     //чистим stdin
}

int read_shot(char *buf) {
    char *read_buf = malloc(READ_BUF_LEN);
    size_t len = READ_BUF_LEN;

    printf("стреляю: ");
    fflush(stdout);
    while(1) {
        ssize_t res = getline(&read_buf, &len, stdin);
        if (read_buf[0] >= 'a' && read_buf[0] <= 'j' && read_buf[1] >= '1' && read_buf[1] <= '9' &&
read_buf[2] == '\n') {
            buf[1] = read_buf[0];
            buf[2] = read_buf[1];
            free(read_buf);
            return 0;
        } else if (read_buf[0] >= 'a' && read_buf[0] <= 'j' && read_buf[1] >= '1' && read_buf[2] >= '0'
&& read_buf[3] == '\n') {
            buf[1] = read_buf[0];
```

```
            buf[2] = 0;
            free(read_buf);
            return 0;
        }
        printf("неверный формат! Повторите ввод\nстреляю: ");
        fflush(stdout);
    }
}

int read_move(char *buf) {
    char read_buf[BUF_LEN];
    size_t len = BUF_LEN;

    while(1) {
        ssize_t res = getline(&buf, &len, stdin);
        if (!strcmp(buf, "мимо\n")) {
            buf[0] = 'M';
            return 0;
        } else if (!strcmp(buf, "ранил\n")) {
            buf[0] = 'R';
            return 1;
        } else if (!strcmp(buf, "убил\n")) {
            buf[0] = 'K';
            return 1;
        } else if (!strcmp(buf, "stop\n")) {
            buf[0] = 'S';
            return 2;
        } else {
            printf("неверный формат! Повторите ввод\nЯ: ");
            fflush(stdout);
        }
    }
}

int move(int fd, char *buf) {
    printf("Противник: ");
    fflush(stdout);
```

```
ssize_t res = read(fd, buf, BUF_LEN);
if (res == 0) {                                 //проверка данных
    printf("\nОшибка передечи данных!\n");
    return -1;
}

if (buf[0] == 'M' || buf[0] == 'N') {
    if (buf[0] == 'M') {
        printf("мимо, ");
    }
    if (buf[2] == 0) {
        printf("стреляю %c10.\nЯ: ", buf[1]);
    } else {
        printf("стреляю %c%c.\nЯ: ", buf[1], buf[2]);
    }
    fflush(stdout);
    memset(buf, '\0', BUF_LEN);

    res = read_move(buf);
    if (res == 0) {
        read_shot(buf);
    } else if (res == 1) {
        buf[1] = 'N';
        buf[2] = 'N';
    } else {
        write(fd, buf, BUF_LEN);
        printf("Игра окончена! Вы проиграли!\n");
        return 1;
    }
} else {
    if (buf[0] == 'S') {
        printf("Игра окончена! Вы победили!\n");
        return 1;
    }
    if (buf[0] == 'R') {
        printf("ранил.\n");
    } else if (buf[0] == 'K') {
```

```
            printf("убил.\n");
        }
        memset(buf, '\0', BUF_LEN);

        printf("Я: ");
        fflush(stdout);
        buf[0] = 'N';
        read_shot(buf);
    }
    write(fd, buf, BUF_LEN);                    //отправляем ход противнику
    memset(buf, '\0', BUF_LEN);                         //чистим буфер
    return 0;
}

int start_server(){                             //создаииие сервера
    int socket_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (socket_fd < 0) {
        return -1;
    }

    struct sockaddr_in srv_addr;
    memset((void *)&srv_addr, '\0', sizeof(srv_addr));
    srv_addr.sin_family = AF_INET;
    srv_addr.sin_addr.s_addr = INADDR_ANY;
    srv_addr.sin_port = htons(SRV_PORT);

    if (bind(socket_fd, (struct sockaddr *) &srv_addr, sizeof(srv_addr)) < 0) {
        close(socket_fd);
        return -1;
    }
    if (listen(socket_fd, 1) < 0) {
        close(socket_fd);
        return -1;
    }
    return socket_fd;
}
```

```
int accept_client(int socket_fd) {                  //принимаем клиента
    struct sockaddr_in client_addr;
    memset((void *)&client_addr, '\0', sizeof(client_addr));
    socklen_t addr_size = sizeof(client_addr);

    printf("Ждем соперника\n");
    int client_fd = accept(socket_fd, (struct sockaddr *)&client_addr, &addr_size);
    if (client_fd < 0) {
        printf("Ошибка подключения\n");
        return -1;
    }
    char *result = inet_ntoa(client_addr.sin_addr);
    printf("%s connected\n", result);
    return client_fd;
}

int server() {                              //функция игры как сервер
    int socket_fd = start_server();
    if (socket_fd < 0) {
        return -1;
    }

    int client_fd = accept_client(socket_fd);
    if (client_fd < 0) {
        return -1;
    }

    char *buf = calloc(BUF_LEN, sizeof(char));
    //flush_stdin();
    while(1) {                                  //цикл игры
        if (move(client_fd, buf) != 0) {
            break;
        }
    }
    free(buf);
    close(client_fd);
    close(socket_fd);
```

```
    return 0;
}

int client() {                          //функция игры как клиент
    printf("Введите ip адрес\n");
    char ip_addr_str[16];
    fgets(ip_addr_str, 16, stdin);

    printf("Введите порт\n");
    int port = 0;
    scanf("%d", &port);

    struct sockaddr_in addr;
    memset((void *)&addr, '\0', sizeof(addr));
    if (inet_aton(ip_addr_str, &addr.sin_addr) < 0) {       //преобразуем адрес
        printf("некорректный адрес\n");
        return -1;
    }
    addr.sin_family = AF_INET;
    addr.sin_port = htons(port);        //преобразуем порт

    int socket_fd = socket(AF_INET, SOCK_STREAM, 0);
    if (socket_fd < 0) {
        return -1;
    }

    if (connect(socket_fd, (struct sockaddr *) &addr, sizeof(addr)) < 0) {
        close(socket_fd);
        printf("ошибка подключения\n");
        return -1;
    }

    char *buf = calloc(BUF_LEN, sizeof(char));
    flush_stdin();
    buf[0] = 'N';
    printf("Я: ");
    fflush(stdout);
```

```
    read_shot(buf);
    write(socket_fd, buf, BUF_LEN);                //отправляем ход противнику
    memset(buf, '\0', BUF_LEN);
    while(1) {                                     //цикл игры
        if (move(socket_fd, buf) != 0) {
            break;
        }
    }
    free(buf);                  //чистка ресурсов
    close(socket_fd);
    return 0;
}

int main(){
    while (1) {
        printf("Создать новую игру или подключиться к сопернику?\n");
        printf("Введите new или connect\n");
        char command[9];
        fgets(command, 9, stdin);              //принимаем опцию игры
        if (!strcmp("new\n", command)) {
            server();                          //запуск игры как сервер
            break;
        } else if(!strcmp("connect\n", command)) {
            client();                          //запуск игры как клиент
            break;
        }
    }
    return 0;
}
```