Федеральное агентство по образованию Российской Федерации

Новгородский государственный университет имени Ярослава Мудрого

Кафедра [Кафедра информационных технологий и систем](http://www.novsu.ru/dept/1238)

**Курсовая работа на тему**

“Многопользовательская сетевая игра «морской бой»”

Отчёт

Выполнил

студент гр. 8091

Васильев Иван Владимирович

«\_\_»\_\_\_\_\_ 2020 г.

Принял

[Владислав Валерьевич Ананьев](http://people.novsu.ru/profiles/html/profileView.do?key=709f5d58-6b4b-463c-b2a3-81ef40ff7202)

«\_\_»\_\_\_\_\_ 2020 г.

Великий Новгород

2020

**Цель работы:**

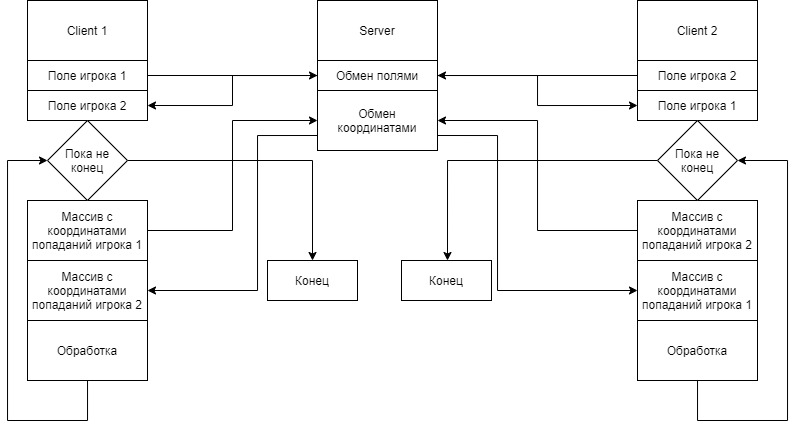
Применить свои знания в области разработки сетевых приложений.

**Задание:**

Написать многопользовательскую сетевую игру «морской бой» с возможностью играть против компьютера в одиночку.

**Разработка проекта:**

Протокол транспортировки: TCP



Правила игры “Морской Бой”

В игре участвуют два игрока, каждый имеет при себе своё поле с кораблями и поле попаданий, суть игры – уничтожить все корабли противников, игра считается проигранной, если ваши все корабли уничтожили.

Игровой процесс:  
Расстановка кораблей:

-в игре есть четыре вида кораблей, различаются по длине и их количестве на поле

1шт- четырёхпалубного (4 клетки)

2шт- трёхпалубного (3 клетки)

3шт-двухпалубного (2 клетки)

4шт-однопалубного (1 клетка)

-корабли не могу стоять вплотную, между ними обязательно должно быть расстояние в одну клетку

Ход игры:

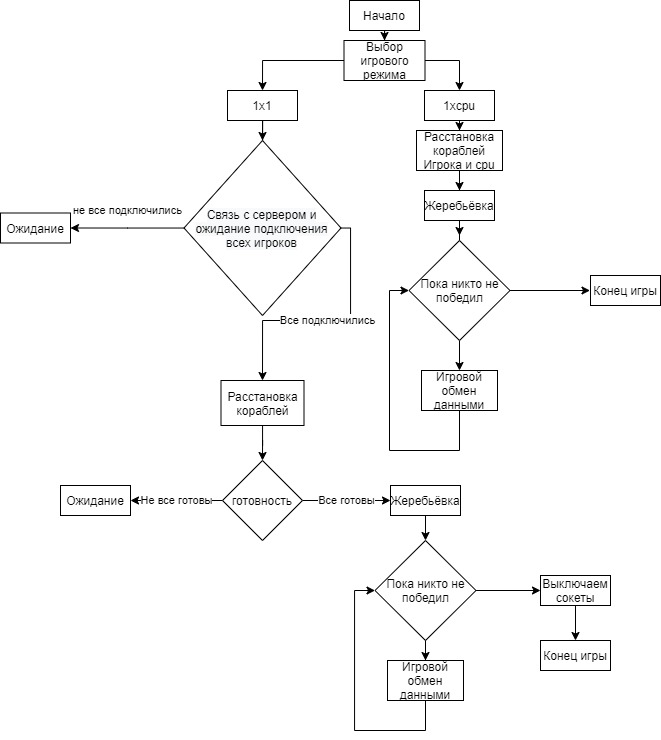
-После расстановки кораблей, игроки вытягивают жребий(случайно) и определяют кто ходит первый

-игрок, кому выпал жребий, ходит первым и называет координаты своего выстрела, если игрок попадает в корабля противника, ему даётся ещё один выстрел, а на карте попаданий регистрируется попадание в корабль, если игрок не попал по кораблю противника, он передаёт ход оппоненту

-игра заканчивается если взорваны все корабли на одном из полей

**Реализация проекта:**

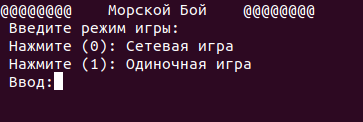
Блок схема игры:



Механика игры:

Главное меню:

1. Выбор игрового режима

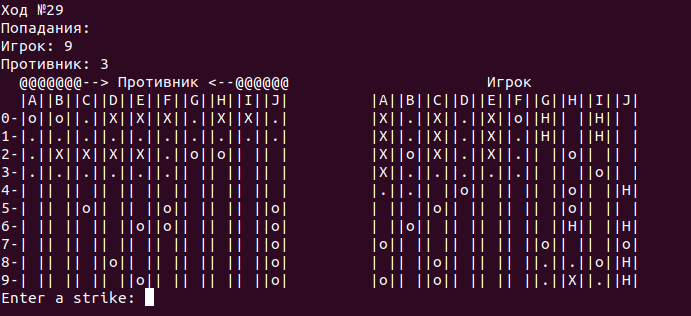


Процесс игры

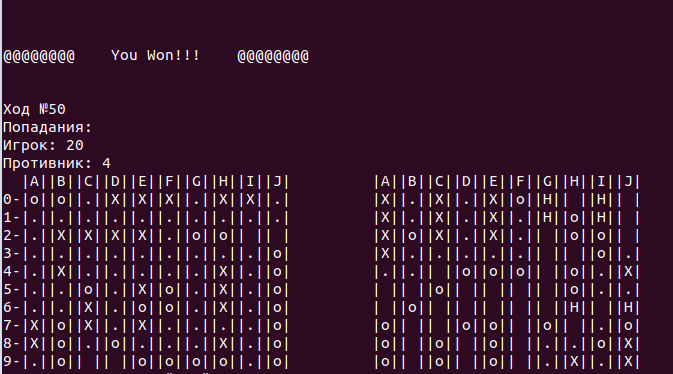
1. Игровой интерфейс состоит из статистики и игровых полей

- Первое поле: противник, опирающееся на заполненный противником поле

- Второе поле: ваше поле, где отображаются ваши корабли и попадания по вам

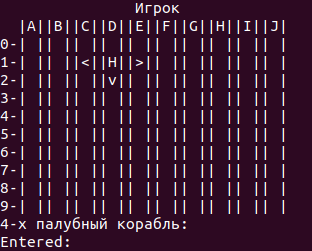


1. Конец игры:

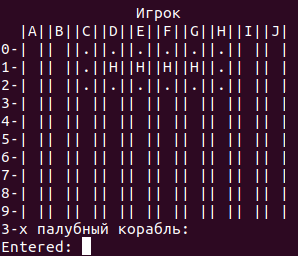


Расстановка кораблей игроком:

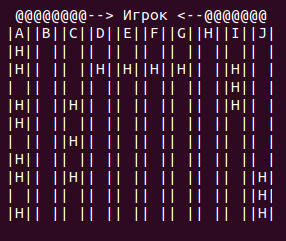
1. Указать координату начало корабля, после этого появляются стрелочки направления продолжения положения корабля, на поле, можно выбрать только стрелочки, все остальные координаты на поле при введение будут признаны как неправильно в ведеными



1. Указать направление (Выбрать одну стрелочку) положения корабля, после чего, он полностью создастся на всю длину, вместе с этим, вокруг корабля создадутся ограничения/подсказки (точки) показывающие игроку, что по правилам, корабли рядом ставить нельзя

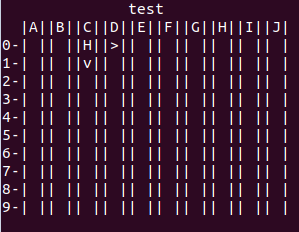


3)После расстановки всех кораблей, поле игрока появится без точек

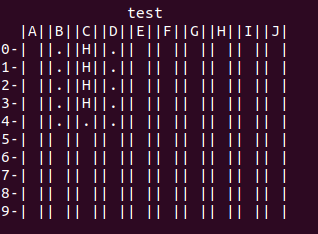


Расстановка кораблей компьютером:

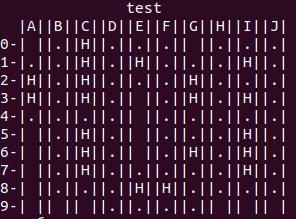
1. Генерацией чисел, выбирается начальная координата корабля



1. Из всех доступных направлений, с помощью генерации выбирается случайное направление спавна



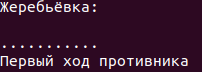
1. Таким образом генерируются все корабли



Жеребьёвка:

-Одиночный режим

После расстановки кораблей, случайным образом определяется кто первым ходит

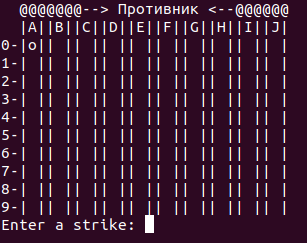


-Многопользовательский режим

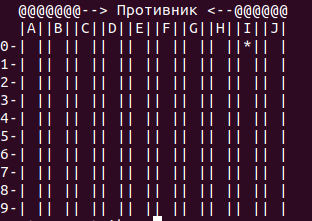
Стрельба игроком:

1. Выбор координаты стрельбы

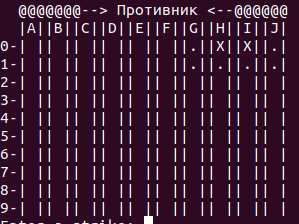
Если вы не попали по вражескому кораблю, поле даст знать о не попадании и ход передаётся оппоненту



Если игрок попадает по кораблю соперника, попадание отображается звёздочкой и даётся ещё один ход



1. Если игрок уничтожает полностью корабль противника, он отображается в виде крестиков, и вокруг него спавнятся точки



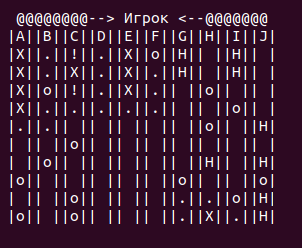
Стрельба компьютером:

1. Компьютер случайным образом выбирает координату, куда будет стрелять, если не попадает, то на поле игрока появляется метка промаха

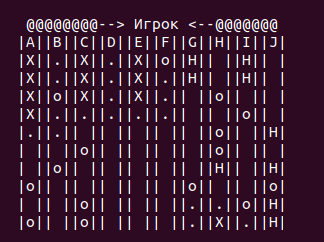
Если попадает, то он отмечает весь корабль восклицательными знаками и шансом с 50% он попадёт и взорвёт его со следующего хода, если взрывает, то корабль уничтожается и

показывается иксами, если промазывает, то следующий ход, он обязательно уничтожит корабль

Обратите внимание на восклицательные знаки:



Следующий ход

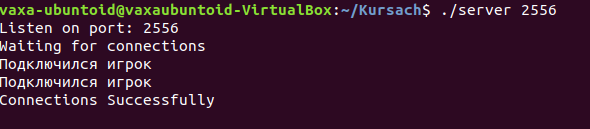


Сервер:

1. После создания сервера, он ожидает двух клиентов



1. Показывает



Функции и что они делают:

void Clear\_Field(Char Field[Size\_Field][Size\_Field] – очищает поле(матрицу) от случайных символов типа char, на вход функции поступает поле

void Hint\_Clear\_Field(char User\_Field[Size\_Field][Size\_Field]) – Вспомогательная функция которая очищает поле от подсказок направлений спавна корабля, убирает стрелочки, на вход поступает поле

void Clear\_User\_Field\_On\_Start\_Game(char User\_Field[Size\_Field][Size\_Field]) –Впомогательная функция, нужна для очистки поля игрока от подсказок(точек), на вход поступает поле

void Show\_Game\_Field\_User(char User\_Field[Size\_Field][Size\_Field]) – Функция которая выводит в консоль поле, нужна для визуализации действий пользователя, на вход поле

void Show\_Game\_Field\_test(char User\_Field[Size\_Field] – Функция с помощью которой я могу показать механики работы других функций

void Move\_Hint(char User\_Field[Size\_Field][Size\_Field], int Row,int Col,int What\_Boat) – Функция направления спавна корабля, играет роль подсказки для пользователя и для упрощёного ввода кораблей пользователем, используется для спавна корабля как пользователем, так и компьютером, на вход поступает поле, строка, колонка и размер корабля

int Check\_Q(char User\_Field[Size\_Field][Size\_Field]) – вспомогательная функция которая ищет на поле стрелочки направлений спавна, на вход поле

return 1 если стрелочки направления есть

return 0 если нету

int Check\_The\_Field(char User\_Field[Size\_Field][Size\_Field],int Row,int Col,int What\_Boat) –Вспомогательная функция для проверки координаты

возвращает 0 если клетка занята, если на поле есть стрелочки но клетка на неё не попадает

возвращает 1 если на поле есть стрелочки и клетка равна стрелочке

возвращает 2 если на поле нету стрелочек

void Spawn\_Full\_Board(char User\_Field[Size\_Field][Size\_Field],int Row,int Col,int What\_Boat)

функция достраиващая по направлению стрелочки корабль до конца

на вход поле, строка, колонка и размер корабля

void Around\_Point(char User\_Field[Size\_Field][Size\_Field],int Row,int Col,int What\_Boat,char Draw) – Выставляет метки вокруг убитого корабля, точки необходимы для реализации правила морского боя, что корабли не могу стоять в платную друг к другу, на вход поле, строка, колонка, размер корабля и метка выставления

int Spawn\_One\_Ship(char User\_Field[Size\_Field][Size\_Field],int What\_Boat) – функция ввода пользователем корабля и установка его

возвращает 1 при успешной установки

возвращает 0 при ошибки ввода

void Spawn\_Boat(char User\_Field[Size\_Field][Size\_Field]) – руководящая функция, выводит информацию о том, какой корабль сейчас устанавливается пользователем в одиночной игре и показывает поле где он устанавливается

void Spawn\_Enemy\_Boat(char Enemy\_Field[Size\_Field][Size\_Field]) –функция расстановки корабле противника в одиночном режиме

int User\_Move(char User\_Field[Size\_Field][Size\_Field],char Enemy\_Field[Size\_Field][Size\_Field],char Hidden\_Field[Size\_Field][Size\_Field]) –функция хода пользователя в одиночном режиме, осуществляет ввод координаты и её обработки

возвращает 1 при попадание

возвращает 0 при промахе

int Exclamation\_Mark(char User\_Field[Size\_Field][Size\_Field]) –функция поиска восклицательных знаков и замены их на попадания с обводкой подбитого корабля метками

возвращает 0 при успешной работе

int Create\_Exclamation(char User\_Field[Size\_Field][Size\_Field],int Row,int Col) – функция создания и замены значка корабля на поле на восклицательный знак, с помощью него в одиночной игре, компьютер выстроит полный корабля если он угадает с направлением корабля

возвращает размер корабля

int CPU\_Move(char User\_Field[Size\_Field][Size\_Field]) – функция хода компьютера в одичном режиме

возвращает 1 при попадании

возвращает 0 при промахе

void Game\_Show(char User\_Field[Size\_Field][Size\_Field],char Hidden\_Field[Size\_Field][Size\_Field],int Hit\_User,int Hit\_CPU,int Motion,int player\_move) – функция вывода игрового интерфейса, выводит поле игрока, поле соперника, статистику

void Start\_Game(char User\_Field[Size\_Field][Size\_Field],char Enemy\_Field[Size\_Field][Size\_Field],char Hidden\_Field[Size\_Field][Size\_Field]) – Главная функция игрового процесса в одиночку, включает в себя жеребьёвку, и смене хода, выводит данные о завершении игры

void Solo\_Games\_Mode() – Начальная функция режима игры в одиночку

int Field\_End(char Field[Size\_Field][Size\_Field]) – функция регистрации игровой статистики в многопользовательском режиме, считывает попадания и возвращает число попаданий

void \*Start\_Coop(void \*data\_input) – поток, использующийся для всего игрового процесса многопользовательской игры

void Multiplayer\_Games\_Mode(int argv,char \*argc[]) – Начальная фунция многопользовательского режима игры, включающий в себя создание сокета и процесса многопользовательской игры

int Select\_Game\_mode() – функция выбора игрового режима пользователем

void \*Server(void \*data\_input) – поток с основным функционалом обмена информацией клиента с сервером, и сервера с клиентами

Запуск программы:

gcc server.c –o server

gcc client.c -o client

./server 2556

./client 127.0.0.1:2556

./client 127.0.0.2:2556

**Код:**

client.c:

#include <stdio.h>

#include <locale.h>

#include <stdlib.h>

#include <unistd.h>

#include <time.h>

#include <malloc.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <sys/time.h>

#include <string.h>

#include <pthread.h>

#include <sys/stat.h>

#define Size\_Field 10

#define Edges\_Field ((Row+What\_Boat<Size\_Field) && (Row-What\_Boat>0)&& (Col+What\_Boat<Size\_Field) && (Col-What\_Boat>0))

struct data

{

int sock;

};

void Clear\_Field(char Field[Size\_Field][Size\_Field])

{

for(int i=0;i<Size\_Field;i++)

{

for(int j=0;j<Size\_Field;j++)

{

Field[i][j]=' ';

}

}

}

void Hint\_Clear\_Field(char User\_Field[Size\_Field][Size\_Field])

{

for(int i=0;i<Size\_Field;i++)

{

for(int j=0;j<Size\_Field;j++)

{

if(User\_Field[i][j]=='^'||User\_Field[i][j]=='>'||User\_Field[i][j]=='v'||User\_Field[i][j]=='<') User\_Field[i][j]=' ';

}

}

}

void Clear\_User\_Field\_On\_Start\_Game(char User\_Field[Size\_Field][Size\_Field])

{

for(int i=0;i<Size\_Field;i++)

{

for(int j=0;j<Size\_Field;j++)

{

if(User\_Field[i][j]=='.') User\_Field[i][j]=' ';

}

}

}

void Show\_Game\_Field\_User(char User\_Field[Size\_Field][Size\_Field])

{

printf("\n");

printf(" Игрок \n");

printf(" |A||B||C||D||E||F||G||H||I||J|\n");

for(int i=0;i<Size\_Field;i++)

{

printf("%d-",i);

for(int j=0;j<Size\_Field;j++)

{

printf("|%c|",User\_Field[i][j]);

}

printf("\n");

}

}

void Show\_Game\_Field\_test(char User\_Field[Size\_Field][Size\_Field])

{

printf("\n");

printf(" test \n");

printf(" |A||B||C||D||E||F||G||H||I||J|\n");

for(int i=0;i<Size\_Field;i++)

{

printf("%d-",i);

for(int j=0;j<Size\_Field;j++)

{

printf("|%c|",User\_Field[i][j]);

}

printf("\n");

}

}

void Move\_Hint(char User\_Field[Size\_Field][Size\_Field], int Row,int Col,int What\_Boat)

{

int Free\_Place;

if(Row-What\_Boat+1>=0)

{

Free\_Place=1;

for(int i=1;i<=What\_Boat-1;i++)

{

if(User\_Field[Row-i][Col]=='H'||User\_Field[Row-i][Col]=='.')

{

Free\_Place=0;

}

}

if (Free\_Place==1) User\_Field[Row-1][Col]='^';

}

if(Row+What\_Boat-1<10){

Free\_Place=1;

for(int i=1;i<=What\_Boat;i++)

{

if(User\_Field[Row+i][Col]=='H'||User\_Field[Row+i][Col]=='.')

{

Free\_Place=0;

}

}

if (Free\_Place==1) User\_Field[Row+1][Col]='v';

}

if(Col+What\_Boat-1<10){

Free\_Place=1;

for(int i=1;i<=What\_Boat-1;i++)

{

if(User\_Field[Row][Col+i]=='H'||User\_Field[Row][Col+i]=='.')

{

Free\_Place=0;

}

}

if (Free\_Place==1) User\_Field[Row][Col+1]='>';

}

if(Col-What\_Boat+1>=0){

Free\_Place=1;

for(int i=1;i<=What\_Boat-1;i++)

{

if(User\_Field[Row][Col-i]=='H'||User\_Field[Row][Col-i]=='.')

{

Free\_Place=0;

}

}

if (Free\_Place==1) User\_Field[Row][Col-1]='<';

}

}

int Check\_Q(char User\_Field[Size\_Field][Size\_Field])

{

for(int i=0;i<Size\_Field;i++)

{

for(int j=0;j<Size\_Field;j++)

{

if(User\_Field[i][j]=='^'||User\_Field[i][j]=='>'||User\_Field[i][j]=='v'||User\_Field[i][j]=='<') return 1;

}

}

return 0;

}

int Check\_The\_Field(char User\_Field[Size\_Field][Size\_Field],int Row,int Col,int What\_Boat)

{

if(User\_Field[Row][Col]=='H') return 0;

if(User\_Field[Row][Col]=='.') return 0;

if(Check\_Q(User\_Field)==1)

{

if(User\_Field[Row][Col]=='^'||User\_Field[Row][Col]=='>'||User\_Field[Row][Col]=='v'||User\_Field[Row][Col]=='<') return 1;

else return 0;

}else return 2;

}

void Spawn\_Full\_Board(char User\_Field[Size\_Field][Size\_Field],int Row,int Col,int What\_Boat)

{

char Direction=User\_Field[Row][Col];

if(Direction=='^')

{

for(int i=0;i<What\_Boat-1;i++)

{

User\_Field[Row-i][Col]='H';

}

}

if(Direction=='v')

{

for(int i=0;i<What\_Boat-1;i++)

{

User\_Field[Row+i][Col]='H';

}

}

if(Direction=='>')

{

for(int i=0;i<What\_Boat-1;i++)

{

User\_Field[Row][Col+i]='H';

}

}

if(Direction=='<')

{

for(int i=0;i<What\_Boat-1;i++)

{

User\_Field[Row][Col-i]='H';

}

}

}

void Around\_Point(char User\_Field[Size\_Field][Size\_Field],int Row,int Col,int What\_Boat,char Draw)

{

int Up=0,Down=0,Right=0,Left=0,i=0;

char Positions;

if(User\_Field[Row-1][Col]=='H' || User\_Field[Row+1][Col]=='H'||User\_Field[Row-1][Col]=='\*'||User\_Field[Row+1][Col]=='\*' || User\_Field[Row+1][Col]=='!'|| User\_Field[Row-1][Col]=='!'|| User\_Field[Row+1][Col]=='X'|| User\_Field[Row-1][Col]=='X') Positions='|';

else if(User\_Field[Row][Col-1]=='H' || User\_Field[Row][Col+1]=='H'||User\_Field[Row][Col-1]=='\*' ||User\_Field[Row][Col+1]=='\*'|| User\_Field[Row][Col+1]=='!'|| User\_Field[Row][Col-1]=='!'|| User\_Field[Row][Col+1]=='X'|| User\_Field[Row][Col-1]=='X') Positions='-';

else if(User\_Field[Row][Col]=='X'||User\_Field[Row][Col]=='H')

{

Positions='\*';

if (Row+1<=9 && User\_Field[Row+1][Col]!='o') User\_Field[Row+1][Col]='.';

if (Row-1>=0 && User\_Field[Row-1][Col]!='o') User\_Field[Row-1][Col]='.';

if (Col+1<=9 && User\_Field[Row][Col+1]!='o') User\_Field[Row][Col+1]='.';

if (Col-1>=0 && User\_Field[Row][Col-1]!='o') User\_Field[Row][Col-1]='.';

}

if(Positions=='|')

{

while(User\_Field[Row+i][Col]=='H'||User\_Field[Row+i][Col]=='\*'||User\_Field[Row+i][Col]=='!'||User\_Field[Row+i][Col]=='X')

{

i++;

continue;

}

if(Row+i >=0 && Row+i <=9 && Col>=0 && Col<=9&&User\_Field[Row+i][Col]!='o') User\_Field[Row+i][Col]=Draw;

i=0;

while(User\_Field[Row+i][Col]=='H'||User\_Field[Row+i][Col]=='\*'||User\_Field[Row+i][Col]=='!'||User\_Field[Row+i][Col]=='X')

{

i--;

continue;

}

if(Row+i >=0 && Row+i <=9 && Col>=0 && Col<=9&&User\_Field[Row+i][Col]!='o') User\_Field[Row+i][Col]=Draw;

while(1)

{

if(User\_Field[Row+Down][Col]=='H'||User\_Field[Row+Down][Col]=='\*'||User\_Field[Row+Down][Col]=='!'||User\_Field[Row+Down][Col]=='X')

{

if(Row+Down >=0 && Row+Down <=9 && Col+1>=0 && Col+1<=9 && User\_Field[Row+Down][Col+1]!='o') User\_Field[Row+Down][Col+1]=Draw;

Down++;

if(Row+Down >=0 && Row+Down <=9 && Col+1>=0 && Col+1<=9 && User\_Field[Row+Down][Col+1]!='o') User\_Field[Row+Down][Col+1]=Draw;

continue;

}else break;

}

Down=0;

while(1)

{

if(User\_Field[Row+Down][Col]=='H'||User\_Field[Row+Down][Col]=='\*'||User\_Field[Row+Down][Col]=='!'||User\_Field[Row+Down][Col]=='X')

{

if(Row+Down >=0 && Row+Down <=9 && Col-1>=0 && Col-1<=9&&User\_Field[Row+Down][Col-1]!='o') User\_Field[Row+Down][Col-1]=Draw;

Down++;

if(Row+Down >=0 && Row+Down <=9 && Col-1>=0 && Col-1<=9&&User\_Field[Row+Down][Col-1]!='o') User\_Field[Row+Down][Col-1]=Draw;

continue;

}else break;

}

Down=0;

while(1)

{

if(User\_Field[Row-Up][Col]=='H'||User\_Field[Row-Up][Col]=='\*'||User\_Field[Row-Up][Col]=='!'||User\_Field[Row-Up][Col]=='X')

{

if(Row-Up >=0 && Row-Up <=9 && Col+1>=0 && Col+1<=9&&User\_Field[Row-Up][Col+1]!='o') User\_Field[Row-Up][Col+1]=Draw;

Up++;

if(Row-Up >=0 && Row-Up <=9 && Col+1>=0 && Col+1<=9&&User\_Field[Row-Up][Col+1]!='o') User\_Field[Row-Up][Col+1]=Draw;

continue;

}else break;

}

Up=0;

while(1)

{

if(User\_Field[Row-Up][Col]=='H'||User\_Field[Row-Up][Col]=='\*'||User\_Field[Row-Up][Col]=='!'||User\_Field[Row-Up][Col]=='X')

{

if(Row-Up >=0 && Row-Up <=9 && Col-1>=0 && Col-1<=9&&User\_Field[Row-Up][Col-1]!='o') User\_Field[Row-Up][Col-1]=Draw;

Up++;

if(Row-Up >=0 && Row-Up <=9 && Col-1>=0 && Col-1<=9&&User\_Field[Row-Up][Col-1]!='o') User\_Field[Row-Up][Col-1]=Draw;

continue;

}else break;

}

Up=0;

}else

{

i=0;

while(User\_Field[Row][Col+i]=='H'||User\_Field[Row][Col+i]=='\*'||User\_Field[Row][Col+i]=='!'||User\_Field[Row][Col+i]=='X')

{

i++;

continue;

}

if(Row >=0 && Row <=9 && Col+i>=0 && Col+i<=9&&User\_Field[Row][Col+i]!='o') User\_Field[Row][Col+i]=Draw;

i=0;

while(User\_Field[Row][Col+i]=='H'||User\_Field[Row][Col+i]=='\*'||User\_Field[Row][Col+i]=='!'||User\_Field[Row][Col+i]=='X')

{

i--;

continue;

}

if(Row >=0 && Row <=9 && Col+i>=0 && Col+i<=9&&User\_Field[Row][Col+i]!='o') User\_Field[Row][Col+i]=Draw;

while(1)

{

if(User\_Field[Row][Col-Left]=='H'||User\_Field[Row][Col-Left]=='\*'||User\_Field[Row][Col-Left]=='!'||User\_Field[Row][Col-Left]=='X')

{

if(Row+1 >=0 && Row+1 <=9 && Col-Left>=0 && Col-Left<=9&&User\_Field[Row+1][Col-Left]!='o') User\_Field[Row+1][Col-Left]=Draw;

Left++;

if(Row+1 >=0 && Row+1 <=9 && Col-Left>=0 && Col-Left<=9&&User\_Field[Row+1][Col-Left]!='o') User\_Field[Row+1][Col-Left]=Draw;

continue;

}else break;

}

Left=0;

while(1)

{

if(User\_Field[Row][Col-Right]=='H'||User\_Field[Row][Col-Right]=='\*'||User\_Field[Row][Col-Right]=='!'||User\_Field[Row][Col-Right]=='X')

{

if(Row-1 >=0 && Row-1 <=9 && Col-Right>=0 && Col-Right<=9&&User\_Field[Row-1][Col-Right]!='o') User\_Field[Row-1][Col-Right]=Draw;

Right++;

if(Row-1 >=0 && Row-1 <=9 && Col-Right>=0 && Col-Right<=9&&User\_Field[Row-1][Col-Right]!='o') User\_Field[Row-1][Col-Right]=Draw;

continue;

}else break;

}

Right=0;

while(1)

{

if(User\_Field[Row][Col+Left]=='H'||User\_Field[Row][Col+Left]=='\*'||User\_Field[Row][Col+Left]=='!'||User\_Field[Row][Col+Left]=='X')

{

if(Row+1 >=0 && Row+1 <=9 && Col+Left>=0 && Col+Left<=9&&User\_Field[Row+1][Col+Left]!='o') User\_Field[Row+1][Col+Left]=Draw;

Left++;

if(Row+1 >=0 && Row+1 <=9 && Col+Left>=0 && Col+Left<=9&&User\_Field[Row+1][Col+Left]!='o') User\_Field[Row+1][Col+Left]=Draw;

continue;

}else break;

}

Left=0;

while(1)

{

if(User\_Field[Row][Col+Right]=='H'||User\_Field[Row][Col+Right]=='\*'||User\_Field[Row][Col+Right]=='!'||User\_Field[Row][Col+Right]=='X')

{

if(Row-1 >=0 && Row-1 <=9 && Col+Right>=0 && Col+Right<=9&&User\_Field[Row-1][Col+Right]!='o') User\_Field[Row-1][Col+Right]=Draw;

Right++;

if(Row-1 >=0 && Row-1 <=9 && Col+Right>=0 && Col+Right<=9&&User\_Field[Row-1][Col+Right]!='o') User\_Field[Row-1][Col+Right]=Draw;

continue;

}else break;

}

Right=0;

}

}

int Spawn\_One\_Ship(char User\_Field[Size\_Field][Size\_Field],int What\_Boat)

{

int Row=0;

int Col;

char s[2];

printf("Entered: ");

gets(s);

Col=s[0]-65;

Row=s[1]-48;

if (Row<=9&&Row>=0&&Col<=9&&Col>=0)

{

if(Check\_The\_Field(User\_Field,Row,Col,What\_Boat)==1 && What\_Boat!=1)

{

Spawn\_Full\_Board(User\_Field,Row,Col,What\_Boat);

Around\_Point(User\_Field,Row,Col,What\_Boat,'.');

Hint\_Clear\_Field(User\_Field);

}else if(Check\_The\_Field(User\_Field,Row,Col,What\_Boat)==2)

{

User\_Field[Row][Col]='H';

if(What\_Boat!=1)

{

Move\_Hint(User\_Field,Row,Col,What\_Boat);

}else

{

Around\_Point(User\_Field,Row,Col,What\_Boat,'.');

}

}else if(Check\_The\_Field(User\_Field,Row,Col,What\_Boat)==0)

{

printf("You cant put it here\n");

return 0;

}

}else

{

printf("You entered it incorrectly\n Example A0 or G4\n");

return 0;

}

return 1;

}

void Spawn\_Boat(char User\_Field[Size\_Field][Size\_Field])

{

printf(" Растановка Кораблей");

sleep(1);

int i=0;

int What\_Boat=4;

while(i!=16){

Show\_Game\_Field\_User(User\_Field);

if(i<2)

{

printf("4-х палубный корабль: \n");

if (Spawn\_One\_Ship(User\_Field,What\_Boat))

{

i++;

continue;

}else continue;

}

if(i<6)

{

printf("3-х палубный корабль:\n");

What\_Boat=3;

if (Spawn\_One\_Ship(User\_Field,What\_Boat))

{

i++;

continue;

}else

{

continue;

}

}

if(i<12)

{

printf("2-х палубный корабль:\n");

What\_Boat=2;

if (Spawn\_One\_Ship(User\_Field,What\_Boat))

{

i++;

continue;

}else continue;

}

if(i<16)

{

printf("1-х палубный корабль:\n");

What\_Boat=1;

if (Spawn\_One\_Ship(User\_Field,What\_Boat))

{

i++;

continue;

}else continue;

}

}

}

void Spawn\_Enemy\_Boat(char Enemy\_Field[Size\_Field][Size\_Field])

{

int key;

int Col,Row,What\_Boat,Num;

srand(time(NULL));

for(int i=0;i<1;i++)

{

key=1;

What\_Boat=4;

do{

Col=rand()%9;

Row=rand()%9;

printf("\nCoord:(%i,%i)\n",Row,Col);//

}while((Enemy\_Field[Row][Col]=='H') || (Enemy\_Field[Row][Col]=='.') );

Enemy\_Field[Row][Col]='H';

Move\_Hint(Enemy\_Field,Row,Col,What\_Boat);

while(key){

Num=rand()%3;

if(Num==0 && Enemy\_Field[Row-1][Col]=='^'&&Row-4>=0)

{

Spawn\_Full\_Board(Enemy\_Field,Row-1,Col,What\_Boat);

Around\_Point(Enemy\_Field,Row-1,Col,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==1 && Enemy\_Field[Row][Col+1]=='>'&& Col+4<=9)

{

Spawn\_Full\_Board(Enemy\_Field,Row,Col+1,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col+1,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==2 && Enemy\_Field[Row+1][Col]=='v'&&Row+4<=9)

{

Spawn\_Full\_Board(Enemy\_Field,Row+1,Col,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==3 && Enemy\_Field[Row][Col-1]=='<'&&Col-4>=0)

{

Spawn\_Full\_Board(Enemy\_Field,Row,Col-1,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col-1,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

}

}

Show\_Game\_Field\_test(Enemy\_Field);

for(int i=0;i<2;i++)

{

key=1;

What\_Boat=3;

do{

Col=rand()%9;

Row=rand()%9;

printf("\nCoord:(%i,%i)\n",Row,Col);//

}while((Enemy\_Field[Row][Col]=='H') || (Enemy\_Field[Row][Col]=='.'));

Enemy\_Field[Row][Col]='H';

Move\_Hint(Enemy\_Field,Row,Col,What\_Boat);

while(key){

Num=rand()%3;

if(Num==0 && Enemy\_Field[Row-1][Col]=='^'&&Row-3>=0)

{

Spawn\_Full\_Board(Enemy\_Field,Row-1,Col,What\_Boat);

Around\_Point(Enemy\_Field,Row-1,Col,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==1 && Enemy\_Field[Row][Col+1]=='>'&&Col+3<=9)

{

Spawn\_Full\_Board(Enemy\_Field,Row,Col+1,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col+1,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==2 && Enemy\_Field[Row+1][Col]=='v'&&Row+3<=9)

{

Spawn\_Full\_Board(Enemy\_Field,Row+1,Col,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==3 && Enemy\_Field[Row][Col-1]=='<'&&Col-3>=0)

{

Spawn\_Full\_Board(Enemy\_Field,Row,Col-1,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col-1,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

}

}

Show\_Game\_Field\_test(Enemy\_Field);

for(int i=0;i<3;i++)

{

key=1;

What\_Boat=2;

do{

Col=rand()%9;

Row=rand()%9;

printf("\nCoord:(%i,%i)\n",Row,Col); //

}while((Enemy\_Field[Row][Col]=='H') || (Enemy\_Field[Row][Col]=='.'));

Enemy\_Field[Row][Col]='H';

Move\_Hint(Enemy\_Field,Row,Col,What\_Boat);

while(key){

Num=rand()%3;

if(Num==0 && Enemy\_Field[Row-1][Col]=='^'&&Row-2>=0)

{

Spawn\_Full\_Board(Enemy\_Field,Row-1,Col,What\_Boat);

Around\_Point(Enemy\_Field,Row-1,Col,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==1 && Enemy\_Field[Row][Col+1]=='>'&&Col+2<=9)

{

Spawn\_Full\_Board(Enemy\_Field,Row,Col+1,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col+1,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==2 && Enemy\_Field[Row+1][Col]=='v'&&Row+2<=9)

{

Spawn\_Full\_Board(Enemy\_Field,Row+1,Col,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

if(Num==3 && Enemy\_Field[Row][Col-1]=='<'&&Col-2>=0)

{

Spawn\_Full\_Board(Enemy\_Field,Row,Col-1,What\_Boat);

Around\_Point(Enemy\_Field,Row,Col-1,What\_Boat,'.');

Hint\_Clear\_Field(Enemy\_Field);

key=0;

}

}

}

Show\_Game\_Field\_test(Enemy\_Field);

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

{

key=1;

What\_Boat=1;

do{

Col=rand()%9;

Row=rand()%9;

printf("\nCoord:(%i,%i)\n",Row,Col);//

}while((Enemy\_Field[Row][Col]=='H') || (Enemy\_Field[Row][Col]=='.') );

Enemy\_Field[Row][Col]='H';

Around\_Point(Enemy\_Field,Row,Col,What\_Boat,'.');

}

Show\_Game\_Field\_test(Enemy\_Field);

}

int User\_Move(char User\_Field[Size\_Field][Size\_Field],char Enemy\_Field[Size\_Field][Size\_Field],char Hidden\_Field[Size\_Field][Size\_Field])

{

int Col,Row,i=1,j=1,What\_Boat;

char s[2];

while(1)

{

printf("Enter a strike: ");

gets(s);

Col=s[0]-65;

Row=s[1]-48;

if (Row<=9&&Row>=0&&Col<=9&&Col>=0&&Enemy\_Field[Row][Col]!='X'&&Enemy\_Field[Row][Col]!='o'&&Enemy\_Field[Row][Col]!='\*'&&Hidden\_Field[Row][Col]!='X'&&Hidden\_Field[Row][Col]!='o'&&Hidden\_Field[Row][Col]!='\*'&&Hidden\_Field[Row][Col]!='.')

{

if(Enemy\_Field[Row][Col]=='H')

{

if(Enemy\_Field[Row+1][Col]=='H' || Enemy\_Field[Row-1][Col]=='H' || Enemy\_Field[Row][Col+1]=='H' || Enemy\_Field[Row][Col-1]=='H') //ПРОВЕРКА НА БОЛЕЕ ЧЕМ ОДНУ ПАЛУБУ

{

Hidden\_Field[Row][Col]='\*';

if(Hidden\_Field[Row+1][Col]=='\*' || Hidden\_Field[Row-1][Col]=='\*')

{

while(Enemy\_Field[Row+i][Col]!='.'&&Row+i<=9)

{

if(Enemy\_Field[Row+i][Col]=='H'&& Hidden\_Field[Row+i][Col]=='\*')

{

i++;

}

else return 1;

}

while(Enemy\_Field[Row-j][Col]!='.'&&Row-j>=0)

{

if(Enemy\_Field[Row-j][Col]=='H'&& Hidden\_Field[Row-j][Col]=='\*') j++;

else return 1;

}

What\_Boat=i+j;

Around\_Point(Hidden\_Field,Row,Col,What\_Boat,'.');

for(int k=0;k<i;k++) Hidden\_Field[Row+k][Col]='X';

for(int k=0;k<j;k++) Hidden\_Field[Row-k][Col]='X';

return 1;

}

i=1;

j=1;

if(Hidden\_Field[Row][Col+1]=='\*' || Hidden\_Field[Row][Col-1]=='\*')

{

while(Enemy\_Field[Row][Col+i]!='.'&&Col+i<=9)

{

if(Enemy\_Field[Row][Col+i]=='H'&& Hidden\_Field[Row][Col+i]=='\*') i++;

else return 1;

}

while(Enemy\_Field[Row][Col-j]!='.'&&Col-j>=0)

{

if(Enemy\_Field[Row][Col-j]=='H'&& Hidden\_Field[Row][Col-j]=='\*') j++;

else return 1;

}

What\_Boat=i+j;

Around\_Point(Hidden\_Field,Row,Col,What\_Boat,'.');

for(int k=0;k<i;k++) Hidden\_Field[Row][Col+k]='X';

for(int k=0;k<j;k++) Hidden\_Field[Row][Col-k]='X';

return 1;

}

return 1;

}else

{

Hidden\_Field[Row][Col]='X';

Around\_Point(Hidden\_Field,Row,Col,1,'.');

return 1;

}

}else

{

Hidden\_Field[Row][Col]='o';

return 0;

}

}else

{

printf("You entered error\n");

}

}

}

int Exclamation\_Mark(char User\_Field[Size\_Field][Size\_Field])

{

int What\_Boat=1,i,j,Row,Col,key=0;

for(i=0;i<Size\_Field;i++)

{

for(j=0;j<Size\_Field;j++)

{

if(User\_Field[i][j]=='!')

{

key=1;

What\_Boat++;

}

}

}

for(i=0;i<Size\_Field;i++)

{

for(j=0;j<Size\_Field;j++)

{

if(User\_Field[i][j]=='!')

{

User\_Field[i][j]='X';

Row=i;

Col=j;

}

}

}

if(key)

{

Around\_Point(User\_Field,Row,Col,What\_Boat,'.');

return What\_Boat-1;

}

return 0;

}

int Create\_Exclamation(char User\_Field[Size\_Field][Size\_Field],int Row,int Col)

{

int i=1,j=0;

while(User\_Field[Row-i][Col]=='H'&&Row-i>=0)

{

User\_Field[Row-i][Col]='!';

i++;

j++;

}

i=1;

while(User\_Field[Row+i][Col]=='H'&&Row+i<=9)

{

User\_Field[Row+i][Col]='!';

i++;

j++;

}

i=1;

while(User\_Field[Row][Col+i]=='H'&&Col<=9)

{

User\_Field[Row][Col+i]='!';

i++;

j++;

}

i=1;

while(User\_Field[Row][Col-i]=='H'&&Col>=0)

{

User\_Field[Row][Col-i]='!';

i++;

j++;

}

return j;

}

int CPU\_Move(char User\_Field[Size\_Field][Size\_Field])

{

if(Exclamation\_Mark(User\_Field)) return 1;

int Col,Row;

int Chek,k=0,p=0;

while(1)

{

srand(time(NULL));

int \*Empty\_Cell=(int\*)malloc(100-20);

{

for(int i=0;i<k;i++) Empty\_Cell[i]=0;

}

for(int i=0;i<Size\_Field;i++)

{

for(int j=0;j<Size\_Field;j++)

{

if(User\_Field[i][j]==' '||User\_Field[i][j]=='H')

{

Empty\_Cell[k]=i+1;

Empty\_Cell[k+1]=j+1;

k+=2;

}

}

}

p=k-1;

k=rand() % p;

if(k%2==0)

{

Row=Empty\_Cell[k];

Col=Empty\_Cell[k+1];

Row-=1;

Col-=1;

}else

{

Col=Empty\_Cell[k];

Row=Empty\_Cell[k-1];

Row-=1;

Col-=1;

}

free(Empty\_Cell);

if(User\_Field[Row][Col]=='H')

{

User\_Field[Row][Col]='X';

Create\_Exclamation(User\_Field,Row,Col);

if(User\_Field[Row+1][Col]=='H'||User\_Field[Row-1][Col]=='H'||User\_Field[Row][Col+1]=='H'||User\_Field[Row][Col-1]=='H')

{

while(1)

{

Chek=rand()%3;

if(Chek==0)

{

if(Row-1>=0)

{

if(User\_Field[Row-1][Col]==' '|| User\_Field[Row-1][Col]=='o')

{

User\_Field[Row-1][Col]='o';

return 0;

}

if(User\_Field[Row-1][Col]=='!')

{

User\_Field[Row-1][Col]='X';

return 1;

}

}else continue;

}

if(Chek==1)

{

if(Col+1<=9)

{

if(User\_Field[Row][Col+1]==' '||User\_Field[Row][Col+1]=='o')

{

User\_Field[Row][Col+1]='o';

return 0;

}

if(User\_Field[Row][Col+1]=='!')

{

User\_Field[Row][Col+1]='X';

return 1;

}

}else continue;

}

if(Chek==2)

{

if(Row+1<=9)

{

if(User\_Field[Row+1][Col]==' '||User\_Field[Row+1][Col]=='o')

{

User\_Field[Row+1][Col]='o';

return 0;

}

if(User\_Field[Row+1][Col]=='!')

{

User\_Field[Row+1][Col]='X';

return 1;

}

}else continue;

}

if(Chek==3)

{

if(Col-1>=0)

{

if(User\_Field[Row][Col-1]==' '||User\_Field[Row][Col-1]=='o')

{

User\_Field[Row][Col-1]='o';

return 0;

}

if(User\_Field[Row][Col-1]=='!')

{

User\_Field[Row][Col-1]='X';

return 1;

}

}else continue;

}

}

}else

{

Around\_Point(User\_Field,Row,Col,1,'.');

return 1;

}

}else if(User\_Field[Row][Col]=='o') continue;

else

{

User\_Field[Row][Col]='o';

return 0;

}

}

}

void Game\_Show(char User\_Field[Size\_Field][Size\_Field],char Hidden\_Field[Size\_Field][Size\_Field],int Hit\_User,int Hit\_CPU,int Motion,int player\_move)

{

printf("Ход №%d\n",Motion);

printf("Попадания: \n");

printf("Игрок: %d\n",Hit\_User);

printf("Противник: %d\n",Hit\_CPU);

if(player\_move==0) printf(" @@@@@@@--> Противник <--@@@@@@ Игрок \n");

else if(player\_move==1)printf(" Противник @@@@@@@@--> Игрок <--@@@@@@@ \n");

printf(" |A||B||C||D||E||F||G||H||I||J| |A||B||C||D||E||F||G||H||I||J|\n");

for(int i=0;i<Size\_Field;i++)

{

printf("%d-",i);

for(int j=0;j<Size\_Field;j++)

{

printf("|%c|",Hidden\_Field[i][j]);

}

printf(" ");

for(int j=0;j<Size\_Field;j++)

{

printf("|%c|",User\_Field[i][j]);

}

printf("\n");

}

}

void Start\_Game(char User\_Field[Size\_Field][Size\_Field],char Enemy\_Field[Size\_Field][Size\_Field],char Hidden\_Field[Size\_Field][Size\_Field])

{

srand(time(NULL));

int player\_move,Hit\_User=0,Hit\_CPU=0,Boat=0,Motion=1;

printf("Жеребьёвка: \n");

printf("\n........... \n");

player\_move=rand() % 2;

sleep(3);

if(player\_move) printf("Первый ход противника\n");

else printf("Первый ход игрока\n");

while((Hit\_User!=20 || Hit\_CPU==20)&&(Hit\_User==20 || Hit\_CPU!=20))

{

Game\_Show(User\_Field,Enemy\_Field,Hit\_User,Hit\_CPU,Motion,player\_move);

Game\_Show(User\_Field,Hidden\_Field,Hit\_User,Hit\_CPU,Motion,player\_move);

sleep(2);

Motion++;

if(player\_move ==0)

{

if(User\_Move(User\_Field,Enemy\_Field,Hidden\_Field) ==1)

{

player\_move=0;

Hit\_User++;

}

else

{

player\_move=1;

}

}

if(player\_move ==1)

{

Boat=0;

if(CPU\_Move(User\_Field))

{

Boat=CPU\_Move(User\_Field);

player\_move=1;

Hit\_CPU+=Boat;

}

else

{

player\_move=0;

}

}

}

if (Hit\_User==20)

{

printf("\n\n\n@@@@@@@@ You Won!!! @@@@@@@@\n\n\n");

Game\_Show(User\_Field,Hidden\_Field,Hit\_User,Hit\_CPU,Motion,2);

}

if (Hit\_CPU==20)

{

printf("\n\n\n@@@@@@@@ You lost((( @@@@@@@@\n\n\n\n");

Game\_Show(User\_Field,Hidden\_Field,Hit\_User,Hit\_CPU,Motion,2);

}

}

void Solo\_Games\_Mode()

{

char User\_Field[Size\_Field][Size\_Field];

char Enemy\_Field[Size\_Field][Size\_Field];

char Hidden\_Field[Size\_Field][Size\_Field];

printf(" Одиночная игра\n");

Clear\_Field(Hidden\_Field);

Clear\_Field(User\_Field);

Clear\_Field(Enemy\_Field);

Spawn\_Boat(User\_Field);

Clear\_User\_Field\_On\_Start\_Game(User\_Field);

Spawn\_Enemy\_Boat(Enemy\_Field);

Start\_Game(User\_Field,Enemy\_Field,Hidden\_Field);

}

int Socket(int domain,int type,int protocol);

void Bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

void Connect(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

void Inet\_pton(int af, const char \*src,void \*dst);

int Field\_End(char Field[Size\_Field][Size\_Field])

{

int Ret=0;

for(int i=0;i<10;i++)

{

for(int j=0;j<10;j++)

{

if(Field[i][j]=='X'||Field[i][j]=='\*') Ret++;

}

}

return Ret;

}

void \*Start\_Coop(void \*data\_input)

{

struct data \*input=(struct data \*)data\_input;

char s[2];

int Col,Row,What\_Boat,i=1,j=1,k=0;

char User\_Field[Size\_Field][Size\_Field];

char Enemy\_Field[Size\_Field][Size\_Field];

char Hidden\_Field[Size\_Field][Size\_Field];

char player\_move;

Clear\_Field(Hidden\_Field);

Clear\_Field(User\_Field);

Clear\_Field(Enemy\_Field);

printf("Жеребьёвка: \n");

printf("\nОжидание игрока... \n");

read(input->sock,&player\_move,sizeof(char));

printf("Move: %c\n",player\_move);

int Your\_Hit,Enemy\_Hit,move=0,popal,exit=0;

int Send\_Coord[40];

if(player\_move =='1')

{

printf("Вы ходите первым\n");

Spawn\_Boat(User\_Field);

Clear\_Field(Hidden\_Field);

send(input->sock,User\_Field,sizeof(char)\*100,0);

Clear\_User\_Field\_On\_Start\_Game(User\_Field);

printf("Ожидание второго игрока\n");

recv(input->sock,Enemy\_Field,sizeof(char)\*100,0);

while(((Enemy\_Hit=Field\_End(User\_Field))!=20)&&((Your\_Hit=Field\_End(Hidden\_Field))!=20))

{

k=0;

popal=1;

move++;

while(popal)

{

exit=0;

i=1;

j=1;

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,1);

printf("Введите координаты выстрела: ");

gets(s);

Col=s[0]-65;

Row=s[1]-48;

if (Row<=9&&Row>=0&&Col<=9&&Col>=0&&Enemy\_Field[Row][Col]!='X'&&Enemy\_Field[Row][Col]!='o'&&Enemy\_Field[Row][Col]!='\*'&&Hidden\_Field[Row][Col]!='X'&&Hidden\_Field[Row][Col]!='o'&&Hidden\_Field[Row][Col]!='\*'&&Hidden\_Field[Row][Col]!='.')

{

Send\_Coord[k]=Row;

Send\_Coord[k+1]=Col;

k+=2;

if(Enemy\_Field[Row][Col]=='H')

{

popal=1;

Your\_Hit++;

if(Enemy\_Field[Row+1][Col]=='H' || Enemy\_Field[Row-1][Col]=='H' || Enemy\_Field[Row][Col+1]=='H' || Enemy\_Field[Row][Col-1]=='H') //ПРОВЕРКА НА БОЛЕЕ ЧЕМ ОДНУ ПАЛУБУ

{

Hidden\_Field[Row][Col]='\*';

if(Hidden\_Field[Row+1][Col]=='\*' || Hidden\_Field[Row-1][Col]=='\*')

{

while(Enemy\_Field[Row+i][Col]!='.'&&Row+i<=9)

{

if(Enemy\_Field[Row+i][Col]=='H'&& Hidden\_Field[Row+i][Col]=='\*')

{

i++;

}

else

{

exit=1;

break;

}

}

if(exit==1) continue;

while(Enemy\_Field[Row-j][Col]!='.'&&Row-j>=0)

{

if(Enemy\_Field[Row-j][Col]=='H'&& Hidden\_Field[Row-j][Col]=='\*')

{

j++;

}

else

{

exit=1;

break;

}

}

if(exit==1) continue;

What\_Boat=i+j;

Around\_Point(Hidden\_Field,Row,Col,What\_Boat,'.');

for(int k=0;k<i;k++) Hidden\_Field[Row+k][Col]='X';

for(int k=0;k<j;k++) Hidden\_Field[Row-k][Col]='X';

continue;

}

i=1;

j=1;

if(Hidden\_Field[Row][Col+1]=='\*' || Hidden\_Field[Row][Col-1]=='\*')

{

while(Enemy\_Field[Row][Col+i]!='.'&&Col+i<=9)

{

if(Enemy\_Field[Row][Col+i]=='H'&& Hidden\_Field[Row][Col+i]=='\*') i++;

else

{

exit=1;

break;

}

}

if(exit==1) continue;

while(Enemy\_Field[Row][Col-j]!='.'&&Col-j>=0)

{

if(Enemy\_Field[Row][Col-j]=='H'&& Hidden\_Field[Row][Col-j]=='\*') j++;

else

{

exit=1;

break;

}

}

if(exit==1) continue;

What\_Boat=i+j;

Around\_Point(Hidden\_Field,Row,Col,What\_Boat,'.');

for(int k=0;k<i;k++) Hidden\_Field[Row][Col+k]='X';

for(int k=0;k<j;k++) Hidden\_Field[Row][Col-k]='X';

continue;

}

continue;

}else

{

Hidden\_Field[Row][Col]='X';

Around\_Point(Hidden\_Field,Row,Col,1,'.');

continue;

}

}else

{

Hidden\_Field[Row][Col]='o';

Send\_Coord[k]=10;

popal=0;

break;

}

}else

{

printf("Вы ввели неправильные координатыб пример: A4 или J9\n");

continue;

}

}

move++;

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,0);

printf("Ожидание хода противника...\n");

send(input->sock,Send\_Coord,sizeof(int)\*40,0);

recv(input->sock,Send\_Coord,sizeof(int)\*40,0);

k=0;

while(Send\_Coord[k]!=10)

{

User\_Field[Send\_Coord[k]][Send\_Coord[k+1]]='X';

k+=2;

Enemy\_Hit++;

}

User\_Field[Send\_Coord[k-2]][Send\_Coord[k-1]]='o';

}

if(Enemy\_Hit==20)

{

printf("\n\n\n@@@@@@@@ You lost((( @@@@@@@@\n\n\n\n");

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,2);

}

if(Your\_Hit==20)

{

printf("\n\n\n@@@@@@@@ You Won!!! @@@@@@@@\n\n\n");

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,2);

}

}

else if(player\_move =='0')

{

printf("Вы ходите вторым\n");

Spawn\_Boat(User\_Field);

Clear\_Field(Hidden\_Field);

printf("Ожидание второго игрока\n");

recv(input->sock,Enemy\_Field,sizeof(char)\*100,0);

Game\_Show(User\_Field,Enemy\_Field,Your\_Hit,Enemy\_Hit,move,0);

send(input->sock,User\_Field,sizeof(char)\*100,0);

Clear\_User\_Field\_On\_Start\_Game(User\_Field);

while(((Enemy\_Hit=Field\_End(User\_Field))!=20)&&((Your\_Hit=Field\_End(Hidden\_Field))!=20))

{

move++;

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,0);

printf("Ожидание хода противника...\n");

recv(input->sock,Send\_Coord,sizeof(int)\*40,0);

k=0;

while(Send\_Coord[k]!=10)

{

User\_Field[Send\_Coord[k]][Send\_Coord[k+1]]='X';

k+=2;

Enemy\_Hit++;

}

User\_Field[Send\_Coord[k-2]][Send\_Coord[k-1]]='o';

move++;

Enemy\_Hit=Field\_End(User\_Field);

Your\_Hit=Field\_End(Hidden\_Field);

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,1);

popal=1;

k=0;

while(popal)

{

exit=0;

i=1;

j=1;

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,1);

printf("Введите координаты выстрела: ");

gets(s);

Col=s[0]-65;

Row=s[1]-48;

if (Row<=9&&Row>=0&&Col<=9&&Col>=0&&Enemy\_Field[Row][Col]!='X'&&Enemy\_Field[Row][Col]!='o'&&Enemy\_Field[Row][Col]!='\*'&&Hidden\_Field[Row][Col]!='X'&&Hidden\_Field[Row][Col]!='o'&&Hidden\_Field[Row][Col]!='\*'&&Hidden\_Field[Row][Col]!='.')

{

Send\_Coord[k]=Row;

Send\_Coord[k+1]=Col;

k+=2;

if(Enemy\_Field[Row][Col]=='H')

{

popal=1;

Your\_Hit++;

if(Enemy\_Field[Row+1][Col]=='H' || Enemy\_Field[Row-1][Col]=='H' || Enemy\_Field[Row][Col+1]=='H' || Enemy\_Field[Row][Col-1]=='H') //ПРОВЕРКА НА БОЛЕЕ ЧЕМ ОДНУ ПАЛУБУ

{

Hidden\_Field[Row][Col]='\*';

if(Hidden\_Field[Row+1][Col]=='\*' || Hidden\_Field[Row-1][Col]=='\*')

{

while(Enemy\_Field[Row+i][Col]!='.'&&Row+i<=9)

{

if(Enemy\_Field[Row+i][Col]=='H'&& Hidden\_Field[Row+i][Col]=='\*')

{

i++;

}

else

{

exit=1;

break;

}

}

if(exit==1) continue;

while(Enemy\_Field[Row-j][Col]!='.'&&Row-j>=0)

{

if(Enemy\_Field[Row-j][Col]=='H'&& Hidden\_Field[Row-j][Col]=='\*')

{

j++;

}

else

{

exit=1;

break;

}

}

if(exit==1) continue;

What\_Boat=i+j;

Around\_Point(Hidden\_Field,Row,Col,What\_Boat,'.');

for(int k=0;k<i;k++) Hidden\_Field[Row+k][Col]='X';

for(int k=0;k<j;k++) Hidden\_Field[Row-k][Col]='X';

continue;

}

i=1;

j=1;

if(Hidden\_Field[Row][Col+1]=='\*' || Hidden\_Field[Row][Col-1]=='\*')

{

while(Enemy\_Field[Row][Col+i]!='.'&&Col+i<=9)

{

if(Enemy\_Field[Row][Col+i]=='H'&& Hidden\_Field[Row][Col+i]=='\*') i++;

else

{

exit=1;

break;

}

}

if(exit==1) continue;

while(Enemy\_Field[Row][Col-j]!='.'&&Col-j>=0)

{

if(Enemy\_Field[Row][Col-j]=='H'&& Hidden\_Field[Row][Col-j]=='\*') j++;

else

{

exit=1;

break;

}

}

if(exit==1) continue;

What\_Boat=i+j;

Around\_Point(Hidden\_Field,Row,Col,What\_Boat,'.');

for(int k=0;k<i;k++) Hidden\_Field[Row][Col+k]='X';

for(int k=0;k<j;k++) Hidden\_Field[Row][Col-k]='X';

continue;

}

continue;

}else

{

Hidden\_Field[Row][Col]='X';

Around\_Point(Hidden\_Field,Row,Col,1,'.');

continue;

}

}else

{

Hidden\_Field[Row][Col]='o';

Send\_Coord[k]=10;

popal=0;

break;

}

}else

{

printf("Вы ввели неправильные координатыб пример: A4 или J9\n");

continue;

}

}

Game\_Show(User\_Field,Enemy\_Field,Your\_Hit,Enemy\_Hit,move,0);

printf("Ожидание хода противника...\n");

send(input->sock,Send\_Coord,sizeof(int)\*40,0);

}

if(Enemy\_Hit==20)

{

printf("\n\n\n@@@@@@@@ You lost((( @@@@@@@@\n\n\n\n");

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,2);

}

if(Your\_Hit==20)

{

printf("\n\n\n@@@@@@@@ You Won!!! @@@@@@@@\n\n\n");

Game\_Show(User\_Field,Hidden\_Field,Your\_Hit,Enemy\_Hit,move,2);

}

}

printf("End send");

return (void \*)(0);

}

void Multiplayer\_Games\_Mode(int argv,char \*argc[])

{

pthread\_t Thread\_Wr,Thread\_Re;

printf(" Сетевая Игра\n");

if (argv != 2)

{

perror("No Address given");

exit(EXIT\_FAILURE);

}

int border=strstr(argc[1],":")-argc[1];

if(border+argc[1]==NULL)

{

perror("Not correct format adress");

exit(EXIT\_FAILURE);

}

char ip[border+1];

strncpy(ip,argc[1],border);

ip[border]='\0';

int port = atoi(argc[1]+border+1);

int sock = Socket(AF\_INET,SOCK\_STREAM,0);

struct sockaddr\_in adr = {0};

adr.sin\_family=AF\_INET;

adr.sin\_port = htons(port);

Inet\_pton(AF\_INET,ip,&adr.sin\_addr);

Connect(sock,(struct sockaddr \*)&adr,sizeof adr);

struct data data\_Thread;

data\_Thread.sock=sock;

pthread\_create(&Thread\_Wr,NULL,Start\_Coop,(void \*)&data\_Thread);

pthread\_join(Thread\_Wr,NULL);

close(sock);

}

int Select\_Game\_mode()

{

int Enter\_Game\_mode;

printf("@@@@@@@@ Морской Бой @@@@@@@@\n Введите режим игры:\n Нажмите (0): Сетевая игра \n Нажмите (1): Одиночная игра\n Ввод:");

scanf("%d",&Enter\_Game\_mode);

return Enter\_Game\_mode;

}

int main(int argv,char \*argc[])

{

setlocale(LC\_ALL,"rus");

int Game\_Mode,Error=1;

while(Error)

{

Game\_Mode = Select\_Game\_mode();

if(Game\_Mode == 1) Solo\_Games\_Mode();

else if(Game\_Mode == 0) Multiplayer\_Games\_Mode(argv,argc);

else

{

printf("You entered the wrong number\n");

continue;

}

}

return 0;

}

int Socket(int domain,int type,int protocol)

{

int res = socket(domain,type,protocol);

if (res == -1){

printf("socket failed");

exit(EXIT\_FAILURE);

}

return res;

}

void Bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen)

{

int res = bind(sockfd,addr,addrlen);

if(res == -1){

printf("Bind failed");

exit(EXIT\_FAILURE);

}

}

void Connect(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen)

{

int res = connect(sockfd,addr,addrlen);

if (res == -1)

{

printf("Connect faild");

exit(EXIT\_FAILURE);

}printf("Подключение успешно\n");

}

void Inet\_pton(int af, const char \*src,void \*dst)

{

int res = inet\_pton(af,src,dst);

if (res == 0){

printf("inet\_pton faild, valid network addres in the specified addres family\n");

exit(EXIT\_FAILURE);

}

if (res == -1)

{

printf("Inet\_pton faild");

exit(EXIT\_FAILURE);

}

}

server.c:

#include <stdio.h>

#include <stdlib.h>

#include <sys/types.h>

#include <sys/socket.h>

#include <netinet/in.h>

#include <arpa/inet.h>

#include <unistd.h>

#include <string.h>

#include <pthread.h>

#include <sys/select.h>

#include <string.h>

#include <time.h>

void Clear\_Field(char Field[10][10])

{

for(int i=0;i<10;i++)

{

for(int j=0;j<10;j++)

{

Field[i][j]=' ';

}

}

}

int Socket(int domain,int type,int protocol);

void Bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen);

void Listen(int sockfd,int backlog);

int Accept(int sockfd, struct sockaddr \*addr,socklen\_t \*addrlen);

int comp(const void \*a,const void \*b);

struct data

{

int \*client\_socket;

socklen\_t \*adrlen;

};

void \*Server(void \*data\_input)

{

printf("Начало игры\n");

struct data \*input=(struct data \*)data\_input;

int valread,i=0,sd,Send\_Coord[2];

char First\_Field[10][10];

char Second\_Field[10][10];

for(int i=0;i<2;i++)

{

sd = input->client\_socket[i];

}

srand(time(NULL));

int player\_move;

player\_move=rand() % 2;

if(player\_move)

{

sd = input->client\_socket[1];

send(sd,"0",sizeof(char),0);

sd = input->client\_socket[0];

send(sd,"1",sizeof(char),0);

sd = input->client\_socket[0];

recv(sd,First\_Field,100,0);

sd = input->client\_socket[1];

send(sd,First\_Field,100,0);

recv(sd,Second\_Field,100,0);

sd = input->client\_socket[0];

send(sd,Second\_Field,100,0);

while(1)

{

sd = input->client\_socket[0];

printf("Получаю\n");

recv(sd,Send\_Coord,sizeof(int)\*40,0);

sd = input->client\_socket[1];

printf("отправляю\n");

send(sd,Send\_Coord,sizeof(int)\*40,0);

printf("Получаю\n");

recv(sd,Send\_Coord,sizeof(int)\*40,0);

sd = input->client\_socket[0];

printf("отправляю\n");

send(sd,Send\_Coord,sizeof(int)\*40,0);

}

}else

{

sd = input->client\_socket[1];

send(sd,"1",sizeof(char),0);

sd = input->client\_socket[0];

send(sd,"0",sizeof(char),0);

sd = input->client\_socket[1];

read(sd,Second\_Field,100);

sd = input->client\_socket[0];

send(sd,Second\_Field,100,0);

read(sd,First\_Field,100);

sd = input->client\_socket[1];

send(sd,First\_Field,100,0);

while(1)

{

sd = input->client\_socket[1];

read(sd,Send\_Coord,sizeof(int)\*40);

sd = input->client\_socket[0];

send(sd,Send\_Coord,sizeof(int)\*40,0);

read(sd,Send\_Coord,sizeof(int)\*40);

sd = input->client\_socket[1];

send(sd,Send\_Coord,sizeof(int)\*40,0);

}

}

}

int main(int argv,char \*argc[])

{

if (argv!=2)

{

perror("Not port given");

exit(EXIT\_FAILURE);

}

int port = atoi(argc[1]);

if (port == 0)

{

perror ("Not Correct port");

exit(EXIT\_FAILURE);

}

pthread\_t thread;

int Master\_Socket = Socket(AF\_INET,SOCK\_STREAM,0); //ipv4 tcp

int New\_Socket1,New\_Socket2;

struct sockaddr\_in adr={0};

if(setsockopt(Master\_Socket,SOL\_SOCKET,SO\_REUSEADDR,(char \*)&adr,sizeof(adr))<0)

{

printf("Error setsockopt");

exit(EXIT\_FAILURE);

}

adr.sin\_family = AF\_INET;

adr.sin\_port = htons(port);

if (adr.sin\_port ==0)

{

printf("The entered port is occupied by another port\n");

printf("New port: %d\n",port);

}else printf("Listen on port: %d \n",port);

int client\_socket[2];

Bind(Master\_Socket,(struct sockaddr \*) &adr,sizeof adr);

Listen(Master\_Socket,2);

socklen\_t adrlen = sizeof adr;

puts("Ожидание подключения");

New\_Socket1=Accept(Master\_Socket,(struct sockaddr \*) &adr, &adrlen);

printf("Подключился игрок\n");

client\_socket[0]=New\_Socket1;

New\_Socket2=Accept(Master\_Socket,(struct sockaddr \*) &adr, &adrlen);

printf("Подключился игрок\n");

client\_socket[1]=New\_Socket2;

puts("Подсключение успешно\n");

struct data threads\_data;

threads\_data.client\_socket=client\_socket;

threads\_data.adrlen=&adrlen;

pthread\_create(&thread,NULL,Server,(void \*)&threads\_data);

pthread\_join(thread,NULL);

close(New\_Socket1);

close(New\_Socket2);

close(Master\_Socket);

return 0;

}

int Socket(int domain,int type,int protocol)

{

int res = socket(domain,type,protocol);

if (res == -1){

printf("socket failed");

exit(EXIT\_FAILURE);

}

return res;

}

void Bind(int sockfd, const struct sockaddr \*addr, socklen\_t addrlen)

{

int res = bind(sockfd,addr,addrlen);

if(res == -1){

printf("Bind failed");

exit(EXIT\_FAILURE);

}

}

void Listen(int sockfd,int backlog)

{

int res = listen(sockfd,backlog);

if (res == -1)

{

printf("listen failed");

exit(EXIT\_FAILURE);

}

}

int Accept(int sockfd, struct sockaddr \*addr,socklen\_t \*addrlen)

{

int res=accept(sockfd,addr,addrlen);

if (res == -1)

{

printf("accept failed");

exit(EXIT\_FAILURE);

}

return res;

}

**Результат работы:**

Программа работает, но есть недочёты

1. При заполнением поля компьютером, возможно зацикливание, если выпавшая случайная координата расположена на единственной свободной точке, где вокруг неё точки и заполненные координаты
2. При быстром вводе координат при выстреле возможен вылет программы, из-за использовании нестабильной функцией gets();

**Вывод**

Задание выполнено не полно, основные механики работают хорошо, но есть недочёты в возможных вылетах из-за описанных выше причин и не очень проработанный интерфейс.