**Морской бой. Пономаренко М.Б.**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-**

#include <iostream> //заголовочный файл с классами, функциями и переменными для организации ввода-вывода в языке программирования C++

#include <time.h> //заглавный файл стандартной библиотеки языка программирования С, содержащий типы и функции для работы с датой и временем.

#include <windows.h> //включает функции WinAPI sleep(), beep() и т.д

#include <conio.h> //библиотека для создания текстового интерфейса пользователя

using namespace std; //пространство имён

enum Color { Black, Blue, Green, Cyan, Red, Magenta, Brown, LightGray, DarkGray, LightBlue = 9, LightGreen, LightCyan, LightRed, LightMagenta, Yellow, White };

// Цвета которыми можно окрашивать: цвет текста; цвет заливки текста

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE); //Дескриптор активного окна

enum Napravlenie { Up = 72, Left = 75, Right = 77, Down = 80, Enter = 13, escepushka = 27, BackSpace = 8, Q = 113 };

//клавиши, обратиться к которым в коде можно через их имя

struct Korabl //структура характеристик кораблика

{

int paluba = { 0 }; //количество палуб

bool horizontal = true; //положение (изначально - горизонтальное)

bool jivoi = true; //проверка на то, унчитожен ли кораблик или нет

int x, y; //Позиции кораблика по "x" и "y" (первая палуба), необходима для правильного уничтожения кораблика

};

void izmenColor(Color text, Color fon); //функция с помощью которой происходит изменение цвета текста

void ustanovCursor(int x, int y); //функция для того чтобы устанавливать позицию курсора в консоли по оси Х и Y

void vzryv(Korabl\* harakteristik, int korabl\_pos[12][24], int plus, int& end); //уничтожение кораблкиков и определение победителя в игре

void PrisvoenPalubKorabl(Korabl\* harakteristika); //присвоение палуб в соответствии с номером кораблика

void avto\_rastanovka(Korabl\* harakteristik, int korabl\_pos[12][24], int plus); //aвтоматическая расстановка корабликов

void RazrisovkaPos(int mas[12][24], int x, int num\_color); //прорисовка игровой карты в соответсвии с положением поставленных корабликов

void Nachalo(); //функция которая будет показана игроку самая первая

void SozdatPole(int skorost, int X, int num\_color); //Функция благодоря которой будет отрисовываться игровое поле

void MenuVyboraRastanovki(char variant[2][30], int T); //меню выбора расстановки

void Repliki(int num);//реплики

int main() //Функция из которой буду вызываться другие и осуществляться инициализация переменых и расстановка корабликов

{

system("chcp 1251"); //Делаем возможным вывод в консоль на русском языке

srand(time(NULL)); //онулирование рандомного значения

char variant[2][30] = {"АВТОМАТИЧЕСКАЯ РАССТАНОВКА","РУЧНАЯ РАССТАНОВКА"};

int korabl\_pos[12][24] = { 0 }; //двумерный массив благодоря которому можно будет отслеживать:

int vrag\_korabl\_pos[12][24] = { 0 }; //двумерный массив благодоря которому можно будет отслеживать:

/\*

0 - свободныем места (куда можно ставить)

1 - позиции возле корабликов (куда нельзя ставить)

2 - место куда попал снаряд но где ничего не было

3 - место где был снаряд и находился кораблик

4 - 1-о палубный кораблик (1-ый)

5 - 1-о палубный кораблик (2-ой)

6 - 1-о палубный кораблик (3-ий)

7 - 1-о палубный кораблик (4-ый)

8 - 2-х палубный кораблик (1-ый)

9 - 2-х палубный кораблик (2-ой)

10 - 2-х палубный кораблик (3-ий)

11 - 3-х палубный кораблик (1-ый)

12 - 3-х палубный кораблик (2-ой)

13 - 4-х палубный кораблик (первый и единственный)

\*/

Korabl\* haracteristic = new Korabl[20]; //создаём 10-ть корабликов характеристики которых описаны в структуре

PrisvoenPalubKorabl(haracteristic); //загрука характеристик корабликов (изначальное положение и количество палуб)

int kluch = 1, y = 2, x = 2, num = 0, obmen = 1, A = 0, xx = 37, yy = 2, up = 0, vniz = 0, nad = 0;

bool ruchnoe\_razmeshen = true, tvoi\_povorot = 0 + rand() % 2, start = false; //переменные для:

/\*

key - переменная для записи нажатой клавиши

start - переменная для проверки на начало боя

speed - скорость прорисовки игрового поля

А - дополнительная переменная для меню

SN - переменная для того чтобы запоминать какой кораблки мы стараемся добить.

up - переменная которая поможет правильно добивать наши кораблики

down - переменная которая поможет правильно добивать наши кораблики

xx - дополнительная перемнная для записи положения курсора на вражеском поле

yy - дополнительная перемнная для записи положения курсора на вражеском поле

y - позиция по Y

x - позиция по X

change - номер расстановки который будет использован

num - номер кораблкиа

manual\_placement - тип расстановки который будет использован

\*/

Nachalo(); //Вывыдим надпись "МОРСКОЙ БОЙ" на экран

system("pause");

//Sleep(1500); //ждём 1.5 секунды

izmenColor(White, Black); //Устанавливаем цвет текста (белый на голубом)

system("cls"); //очищяем консоль

SozdatPole(1, 0, 9); //создаём игровое поле

MenuVyboraRastanovki(variant, obmen);

do //запускаем цикл

{

kluch = \_getch(); //получаем номер нажатой клавиши

switch (kluch) //запускаем свитч

{

case Up: //если нажали "Стрелочка вверх"

if (y > 0) //если "y" больше 0

{

y--; //вычитаем из значение по "y" 1

obmen = y; //выбранный пункт

MenuVyboraRastanovki(variant, obmen); //вывод пунктов на экран

}

break;

case Down: //если нажали "Стрелочка вниз"

if (y < 1) //если "y" меньше 1

{

y++; //прибавляем к значению по "y" 1

obmen = y; //выбранный пункт

MenuVyboraRastanovki(variant, obmen); //вывод пунктов на экран

}

break;

case Enter:

do

{

switch (obmen)

{

case 0: // ксли 0

ruchnoe\_razmeshen = false; //автоматическая расстанвкоа

A++; //чтобы заврешить цикл

break;

case 1:

ruchnoe\_razmeshen = true; //ручная расстановка

A++; //чтобы заврешить цикл

y++; //прибавляем к значению по "y" 1

break;

}

}

while (kluch != Enter);

}

}

while (A < 1);

system("cls"); //очищяем консоль

SozdatPole(0, 0, 9); //создаём игровое поле

x = 2, y = 2;

if (ruchnoe\_razmeshen == false) //если выбрана автоматическая расстановка

{

avto\_rastanovka(haracteristic, korabl\_pos, 0); //передаём необходимые значения для того что автоматически расставить кораблки

RazrisovkaPos(korabl\_pos, 2, 0); //перерисовываем игровую локации с учетём внесенных изменений (поставили кораблик

start = true; //начинаем игру

}

else if (ruchnoe\_razmeshen == true && num < 10) //ручная расстановка

{

do //запускаем цикл в котором при нажатии на ту или иную клавишу будеи делать то или иное дейстивие

{

RazrisovkaPos(korabl\_pos, 2, 0); //перерисовываем игровую локации с учетём внесенных изменений (поставили кораблик)

ustanovCursor(x, y); //ставим курсор на позицию изменяемых ниже переменных (X, Y)

izmenColor(Black, Black); //устанавливаем черный цвет так как таким цветом будет выводиться кораблик

if (haracteristic[num].horizontal == true) //если кораблик имеет горизонтальное положение

{

for (size\_t i = 0; i < haracteristic[num].paluba; i++) //запускаем цикл который будет отрисовыввать кораблик с нужным колличеством палуб

{

cout << "++"; //2 "+" так как одна клеточка в ширину в игре равна 2-ум клеточкам в ширину в консоли

}

}

else if (haracteristic[num].horizontal == false)

{

y += haracteristic[num].paluba; //изменяем позицию по "y" на то сколько палуб у кораблика

for (size\_t i = 0; i < haracteristic[num].paluba; i++) //запускаем цикл который будет отрисовыввать кораблик с нужным колличеством палуб

{

cout << "++"; //2 "+" так как одна клеточка в ширину в игре равна 2-ум клеточкам в ширину в консоли

y--; //спускаемся вниз на одну клеточку

ustanovCursor(x, y); //устанавливаем позицию на одну клеточку ниже и на изначальное по Х

}

}

kluch = \_getch();//функция возвращает номер нажатой клавиши

switch (kluch) //Выполняем различные функции в соответствии с нажатой клавишью

{

case Left: //если нажали "Стрелочка влево"

if (x > 3) //позиция до которой разрешено перемещение в лево

{

x -= 2; //меняем значение по Х на -2 так как одна клеточка в игре равна двум в консоли

}

break;

case Right: //если нажали "Стрелочка вправо"

if (haracteristic[num].horizontal == true) //Если кораблик имеет горизонтальное положение

{

if (x < 22 - haracteristic[num].paluba \* 2) // проверяем не достиг ли он позиции до которой ему можно передвигаться(чтобы он не вышел за границы поля)

{

x += 2; //меняем значение переменной Х на +2 так как одна клеточка в игре равна двум в консоли

}

}

else //Если кораблик имеет вертикальное положение

{

if (x < 20) // проверяем не достиг ли он позиции до которой ему можно передвигаться(чтобы он не вышел за границы поля)

{

x += 2; //меняем значение переменной Х на +2 так как одна клеточка в игре равна двум в консоли

}

}

break; //закрыли этот кейс

case Up: //если нажали "Стрелочка вверх"

if (y > 2) //позиция до которой разрешено перемещение вверх

{

y--; //меняем значение по Y на -1

}

break; //закрыли этот кейс

case Down: //если нажали "Стрелочка вних"

if (haracteristic[num].horizontal == false) //Если кораблик имеет вертикальное положение

{

if (y < 12 - haracteristic[num].paluba) // проверяем не достиг ли он позиции до которой ему можно передвигаться (так как он 4-х палубный, и его нужно ограничивать на 3 клеточки ранее)

{

y++; //меняем значение по Y на +1

}

}

else

{

if (y < 11) // проверяем не достиг ли он позиции до которой ему можно передвигаться(чтобы он не вышел за границы поля)

{

y++; //меняем значение по Y на +1

}

}

break; //закрыли этот кейс

case Q: //если нажали "Q"

haracteristic[num].horizontal = !haracteristic[num].horizontal; //изменяем ориентацию кораблика на противоположную

if (haracteristic[num].horizontal == false && y + haracteristic[num].paluba >= 13)

//проверяем заденит ли кораблик границу по вертикале при перевороте, если его не отодвинуть

{

y -= y + haracteristic[num].paluba - 12; //если да, то заранее изменем его положение чтобы он не прошел сковзь границы поля по вертикали

}

else if (haracteristic[num].horizontal == true && x + haracteristic[num].paluba \* 2 >= 24)

//проверяем заденит ли кораблик границу по вертикале при перевороте, если его не отодвинуть

{

x -= (x + haracteristic[num].paluba \* 2) - 22; //если да, то заранее изменем его положение чтобы он не прошел сковзь границы поля по вертикали

}

for (size\_t i = 0; i < 10 - num; i++) //изменяем ориентацию так же и для последующих корабликов

{

haracteristic[num + i].horizontal = haracteristic[num].horizontal; // заранее изменяем положение следующего кораблика на то которое мы выбрали сейчас

}

break; //закрыли этот кейс

case Enter: //если нажали "Enter"

if (haracteristic[num].horizontal == true && korabl\_pos[y - 1][x] == 0 && korabl\_pos[y - 1][x + haracteristic[num].paluba \* 2 - 2] == 0)

{

haracteristic[num].x = x + 2, haracteristic[num].y = y;

for (size\_t q = 0; q < 3; q++) //выполнеям нижеуказанный код 3 раза так как нам необходимо окупировать кораблик чтобы они не стояли вполтную к друг другу

{

for (size\_t w = 1; w < haracteristic[num].paluba \* 3 + 1; w++) //выполнеям нижеуказанный код 11 раз так как надо чтобы 4-х палубный кораблик полностью окупировался

{

korabl\_pos[y - 2 + q][x - 3 + w] = 1; //меняем значение окупируемых позиций в двумерном массиве который будем прверять на различные цифорки, на "1"

}

}

for (int g = 0; g < 2 \* haracteristic[num].paluba; g++) //выполнеям нижеуказанный код 8 раз так как кораблик 4-х палубный, а одна палуба по горизонатали 2 клеточки

{

korabl\_pos[y - 1][x + g] = 13 - num; //меняем значение позиций в двумерном массиве который будем прверять на различные цифорки, на "2"

}

num++; //переходим к следующему кораблику

}

if (haracteristic[num].horizontal == false && korabl\_pos[y - 1][x] == 0 && korabl\_pos[y + haracteristic[num].paluba - 2][x] == 0)

{

haracteristic[num].x = x + 2, haracteristic[num].y = y;

for (size\_t q = 0; q < haracteristic[num].paluba + 2; q++) //выполнеям нижеуказанный код 6 раз так как нам необходимо окупировать кораблик чтобы они не стояли вполтную к друг другу

{

for (size\_t w = 0; w < 6; w++) //выполнеям нижеуказанный код 6 раз так как нам необходимо окупировать кораблик чтобы они не стояли вполтную к друг другу

{

korabl\_pos[y - 2 + q][x - 2 + w] = 1; //меняем значение окупируемых позиций в двумерном массиве который будем прверять на различные цифорки, на "1"

}

}

for (int g = 0; g < haracteristic[num].paluba; g++) //выполнеям нижеуказанный код 3 раза так как кораблик 3-х палубный

{

for (int n = 0; n < 2; n++) //выполняем нижеуказаныый код 2 раза так как нам нужно поменять занчения в массиве для 2-ух клеточек (одной палубы)

{

korabl\_pos[y - 1 + g][x + n] = 13 - num; //меняем значение позиций в двумерном массиве который будем прверять на различные цифорки, на "2"

}

}

num++; //переходим к следующему кораблику

}

}

start = true;//запускаем игру

} while (num <= 9); //цикл работает до тех пор пока количество расставленных корабликов не равно 10

}

Sleep(20);

SozdatPole(1, 35, 4); //создаём игровое поле

avto\_rastanovka(haracteristic, vrag\_korabl\_pos, 10); //автоматическая расстанвока для вражеских корабликов

RazrisovkaPos(vrag\_korabl\_pos, 37, 15); //отрисовка вражеских кораблей/поля

do

{

if (tvoi\_povorot == true) //если моя очередь атаковать

{

do //запускаем цикл в котором при нажатии на ту или иную клавишу будеи делать то или иное дейстивие

{

x = xx, y = yy; //переменная которая будет запоминать где находился курсор

RazrisovkaPos(vrag\_korabl\_pos, 37, 15); //перерисовываем игровую локации с учетём внесенных изменений (поставили кораблик)

izmenColor(DarkGray, DarkGray); //устанавливаем черный цвет так как таким цветом будет выводиться кораблик

ustanovCursor(x, y); //ставим курсор на позицию изменяемых ниже переменных (X, Y)

cout << "++"; //2 "+" так как одна клеточка в ширину в игре равна 2-ум клеточкам в ширину в консоли

kluch = \_getch();//функция возвращает номер нажатой клавиши

switch (kluch) //Выполняем различные функции в соответствии с нажатой клавишью

{

case Left: //если нажали "Стрелочка влево"

if (x > 38) //позиция до которой разрешено перемещение в лево

{

x -= 2; //меняем значение по Х на -2 так как одна клеточка в игре равна двум в консоли

xx = x; //запоминаем положение курсора

}

break;

case Right: //если нажали "Стрелочка вправо"

if (x < 55) // проверяем не достиг ли он позиции до которой ему можно передвигаться(чтобы он не вышел за границы поля)

{

x += 2; //меняем значение переменной Х на +2 так как одна клеточка в игре равна двум в консоли

xx = x; //запоминаем положение курсора

}

break; //закрыли этот кейс

case Up: //если нажали "Стрелочка вверх"

if (y > 2) //позиция до которой разрешено перемещение вверх

{

y--; //меняем значение по Y на -1

yy = y; //запоминаем положение курсора

}

break; //закрыли этот кейс

case Down: //если нажали "Стрелочка вних"

if (y < 11) //позиция до которой разрешено перемещение вниз

{

y++; //меняем значение по Y на +1

yy = y; //запоминаем положение курсора

}

break;

case Enter:

if (vrag\_korabl\_pos[y - 1][x - 35] >= 4) //проверяем позицию по которой мы выстрельнули на наличие корабликов

{

for (size\_t i = 0; i < 2; i++) //если это так то мызаменяем в клеточке номерок кораблика на "3"

{

vrag\_korabl\_pos[y - 1][x - 35 + i] = 3; // изменяем значение

vzryv(haracteristic, vrag\_korabl\_pos, 10, nad); //проверяем на уничтожение

}

Repliki(9); //вывод реплики на экран

Sleep(500); //небольшая пауза

}

else if (vrag\_korabl\_pos[y - 1][x - 35] < 2) //Если на позиции по которой был произведён огонь ничего нету

{

for (size\_t i = 0; i < 2; i++) //если это так то мызаменяем в клеточке номерок кораблика на "2"

{

vrag\_korabl\_pos[y - 1][x - 35 + i] = 2; // изменяем значение

}

Sleep(500); //небольшая пауза

Repliki(5); //вывод реплики на экран

tvoi\_povorot = false; //очередь атаки противника

}

break;

}

} while (kluch != Enter); //выполнять код который демонстрируется в цикл до тех пор пока не нажата клавиша "Enter"

}

if (tvoi\_povorot == false) //если ход противника

{

Sleep(0 + rand() % 400); //небольшая пауза

Repliki(0); //вывод реплики на экран

RazrisovkaPos(korabl\_pos, 2, 0); //перерисовываем наше поле

Sleep(0 + rand() % 400); //небольшая пауза

x = (2 + rand() % 10) \* 2, y = 2 + rand() % 10; //случайным образом выдаём кооординату по котороой будет происходить атака

if (korabl\_pos[y - 1][x - 1] >= 4) //проверяем позицию по которой мы выстрельнули на наличие корабликов

{

for (size\_t i = 0; i < 2; i++) //если это так то мызаменяем в клеточке номерок кораблика на "3"

{

korabl\_pos[y - 1][x - 2 + i] = 3; // изменяем значение

vzryv(haracteristic, korabl\_pos, 0, nad); //проверяем на уничтожение

}

Repliki(2); //вывод реплики на экран

Sleep(0 + rand() % 750); //небольшая пауза

}

else if (korabl\_pos[y - 1][x - 1] <= 2) //Если на позиции по которой был произведён огонь ничего нету

{

for (size\_t i = 0; i < 2; i++) //если это так то мызаменяем в клеточке номерок кораблика на "2"

{

korabl\_pos[y - 1][x - 2 + i] = 2; //изменяем значение

}

Repliki(1); //вывод реплики на экран

Sleep(0 + rand() % 750); //небольшая пауза

tvoi\_povorot = true; //моя очередь атаки

}

}

} while (nad == 0);

RazrisovkaPos(korabl\_pos, 2, 0);

RazrisovkaPos(vrag\_korabl\_pos, 37, 15); //перерисовываем игровую локации с учетём внесенных изменений (поставили кораблик)

if (nad == 2) //

{

Repliki(7);

}

else

{

Repliki(3);

}

Sleep(5000);

cout << endl;

cout << "(((((((!!!УДАЧИ!!!)))))))" << endl;

//system("pause");

}

void izmenColor(Color text, Color fon) //функция с помощью которой происходит изменение цвета текста

{

SetConsoleTextAttribute(hStdOut, (WORD)((fon << 4) | text)); //установка параметров текста

}

void ustanovCursor(int x, int y) //функция для того чтобы устанавливать позицию курсора в консоли по оси Х и Y

{

COORD myCoords = { x,y }; //инициализация координат

SetConsoleCursorPosition(hStdOut, myCoords); //Способ перемещения курсора на нужные координаты

}

void vzryv(Korabl\* harakteristik, int korabl\_pos[12][24], int plus, int& end) //уничтожение кораблкиков и определение победителя в игре

{

int A = 0; //переменная для правильной зарисовки границы вокруг взоравнного кораблика

int B = 13; //переменная, которая будет помогать указывать на нужный кораблик

int C = 0; //переменная, которая хранит количество уничтоженных корабликов (если = 10, то игра завершается)

bool alive = false; //переменная с помощью которой мы будем проверять уничтожение корабликов

for (size\_t n = 13; n > 3; n--) //проверяем все номерки корабликов на наличие их в массиве

{

for (size\_t i = 1; i < 11; i++) //проганяем массив по строчкам

{

for (size\_t r = 2; r < 22; r++) //проганяем массив по столбцам

{

if (korabl\_pos[i][r] == n) //проверяем, есть ли в массиве хоть один номерок с корабликом

{

alive = true; //в случае успеха - кораблик жив/ранен

i = 11; //меняем значение чтобы перейти к следуещему номерку

break; //выходим из цикла

}

}

}

if (alive == false) //если номерок найден не был, то говорим, что кораблик которому был присвоен этот номер уничтожен

{

harakteristik[B - n + plus].jivoi = false; //говорим что данный кораблик уничтожен

}

else //иначе

{

alive = false; //меняем данную переменную на "false", чтобы проверить следующий кораблик

}

}

for (size\_t i = plus; i < 10 + plus; i++) //запускаем цикл, в котором будем проверять все кораблики одной команды на уничтожение

{

if (i == 0 || i == 10) //для 4-х палубных

{

A = 0; //меняем значение переменной для правильной отрисовки

}

else if (i == 1 || i == 2 || i == 11 || i == 12) //для 3-х палубных

{

A = 1; //меняем значение переменной для правильной отрисовки

}

else if (i > 2 && i < 6 || i > 12 && i < 16) //для 2-х палубных

{

A = 2; //меняем значение переменной для правильной отрисовки

}

else //для 1-о палубных

{

A = 3; //меняем значение переменной для правильной отрисовки

}

if (harakteristik[i].jivoi == false) //если данный кораблик уничтожен

{

if (harakteristik[i].horizontal == true) //если он имеет горизонтальное положение

{

for (size\_t q = 0; q < 3; q++) //выполнеям нижеуказанный код 3 раза так как нам необходимо окупировать кораблик чтобы они не стояли вполтную к друг другу

{

for (size\_t w = 0; w < harakteristik[i].paluba \* 3 + A; w++) //выполнеям нижеуказанный код нужное количество раз (для каждого отдельно считаем), чтобы "окупировать" уничтоженный кораблик

{

korabl\_pos[harakteristik[i].y - 2 + q][harakteristik[i].x - 4 + w] = 2; //меняем значение "окупируемых" позиций в двумерном массиве который будем прверять на различные цифорки, на "2" (промах)

}

}

for (size\_t g = 0; g < 2 \* harakteristik[i].paluba; g++) //выполнеям нижеуказанный код нужное количество раз (для каждого отдельно считаем), чтобы выделить уничтоженный кораблик

{

korabl\_pos[harakteristik[i].y - 1][harakteristik[i].x - 2 + g] = 3; //меняем значение "окупируемых" позиций в двумерном массиве который будем прверять на различные цифорки, на "3" (поподание)

}

}

else if (harakteristik[i].horizontal == false) //если он имеет вертикальное положение

{

for (size\_t q = 0; q < harakteristik[i].paluba + 2; q++) //выполнеям нижеуказанный код 3 раза так как нам необходимо окупировать кораблик чтобы они не стояли вполтную к друг другу

{

for (int w = -4; w < 2; w++) //выполнеям нижеуказанный код нужное количество раз (для каждого отдельно считаем), чтобы "окупировать" уничтоженный кораблик

{

korabl\_pos[harakteristik[i].y - 2 + q][harakteristik[i].x + w] = 2; //меняем значение "окупируемых" позиций в двумерном массиве который будем прверять на различные цифорки, на "2" (промах)

}

}

for (int g = 0; g < harakteristik[i].paluba; g++) //выполнеям нижеуказанный код нужное количество раз (для каждого отдельно считаем), чтобы выделить уничтоженный кораблик

{

for (int n = -2; n < 0; n++) //выполняем нижеуказаныый код 2 раза так как нам нужно поменять значения в массиве для 2-ух клеточек (одной палубы)

{

korabl\_pos[harakteristik[i].y - 1 + g][harakteristik[i].x + n] = 3; //меняем значение "окупируемых" позиций в двумерном массиве который будем прверять на различные цифорки, на "3" (поподание)

}

}

}

C++; // увеличиваем значение счётчика уничтоженных корабликов на 1

}

}

if (C == 10 && plus == 0) //если все наши кораблики уничтожены

{

end = 1; //завершаем игру (поражение)

}

else if (C == 10 && plus == 10) //если все кораблики врага уничтожены

{

end = 2; //завершаем игру (победа)

}

else //иначе

{

C = 0; //онулдируем счетчик уничтоженных корабликов

}

}

void PrisvoenPalubKorabl(Korabl\* harakteristika) //присвоение палуб в соответствии с номером кораблика

{

for (size\_t i = 0; i < 20; i++) //выполнеям нижеуказанный код 20 раз так как общее количество корабликов равно 20

{

if (i == 0 || i == 10) //для 4-х палубных

{

harakteristika[i].paluba = 4; //делаем кораблик с 4-мя палубами

}

else if (i > 0 && i <= 2 || i > 10 && i <= 12) //для 3-х палубных

{

harakteristika[i].paluba = 3; //делаем кораблик с 3-мя палубами

}

else if (i > 2 && i <= 5 || i > 12 && i <= 15) //для 2-х палубных

{

harakteristika[i].paluba = 2; //делаем кораблик с 2-мя палубами

}

else if (i > 5 && i <= 9 || i > 15 && i <= 19) ///для 1-о палубных

{

harakteristika[i].paluba = 1; //делаем кораблик с 1-ой палубой

}

}

}

void avto\_rastanovka(Korabl\* harakteristik, int korabl\_pos[12][24], int plus) //aвтоматическая расстановка корабликов

{

int y, x, ship = plus; //переменная для координат и номера кораблика

while (ship < plus + 10) //выполняем нижеуказанный код до тех пор пока, не будут расставлены все 10 корабликов

{

harakteristik[ship].horizontal = 0 + rand() % 2; //присваиваем рандомную ориентацию

y = 2 + rand() % 9, x = (1 + rand() % 9) \* 2; //присваиваем рандочные координаты (в рамках игровой зоны)

if (harakteristik[ship].horizontal == false && y + harakteristik[ship].paluba >= 13)

//проверяем заденит ли кораблик границу по вертикале при перевороте, если его не отодвинуть

{

y -= y + harakteristik[ship].paluba - 12; //если да, то заранее изменем его положение чтобы он не прошел сковзь границы поля по вертикали

}

else if (harakteristik[ship].horizontal == true && x + harakteristik[ship].paluba \* 2 >= 24)

//проверяем заденит ли кораблик границу по вертикале при перевороте, если его не отодвинуть

{

x -= (x + harakteristik[ship].paluba \* 2) - 22; //если да, то заранее изменем его положение чтобы он не прошел сковзь границы поля по вертикали

}

if (harakteristik[ship].horizontal == true && korabl\_pos[y - 1][x] == 0 && korabl\_pos[y - 1][x + harakteristik[ship].paluba \* 2 - 2] == 0)

// проверемя имеет ли кораблик горизонтальное положение и нет ли ничего на позициях куда он будет устанавливаться

{

harakteristik[ship].x = x + 2, harakteristik[ship].y = y; //запоминаем координаты первой палубы у каждого кораблика, чтобы в дальнейшем правильно их взрывать

for (size\_t q = 0; q < 3; q++) //выполнеям нижеуказанный код 3 раза так как нам необходимо "окупировать" кораблик чтобы они не стояли вполтную к друг другу

{

for (size\_t w = 1; w < harakteristik[ship].paluba \* 3 + 1; w++)//выполнеям нижеуказанный код нужное количество раз (для каждого отдельно считаем), чтобы "окупировать" уничтоженный кораблик

{

korabl\_pos[y - 2 + q][x - 3 + w] = 1; //меняем значение "окупируемых" позиций в двумерном массиве который будем прверять на различные цифорки, на "1" (окупируемая позиция живого кораблика)

}

}

for (int g = 0; g < 2 \* harakteristik[ship].paluba; g++) //выполнеям нижеуказанный код нужное количество раз (для каждого отдельно считаем), чтобы выделить уничтоженный кораблик

{

korabl\_pos[y - 1][x + g] = 13 - ship + plus; //меняем значение позиций в двумерном массиве который будем прверять на различные цифорки, на "4-13" (номера корабликов (у каждого свой))

}

ship++; //переходим к следующему кораблику

}

if (harakteristik[ship].horizontal == false && korabl\_pos[y - 1][x] == 0 && korabl\_pos[y + harakteristik[ship].paluba - 2][x] == 0)

// проверемя имеет ли кораблик горизонтальное положение и нет ли ничего на позициях куда он будет устанавливаться

{

harakteristik[ship].x = x + 2, harakteristik[ship].y = y;//запоминаем координаты первой палубы у каждого кораблика, чтобы в дальнейшем правильно их взрывать

for (size\_t q = 0; q < harakteristik[ship].paluba + 2; q++)

{

for (size\_t w = 0; w < 6; w++)

{

korabl\_pos[y - 2 + q][x - 2 + w] = 1; //меняем значение "окупируемых" позиций в двумерном массиве который будем прверять на различные цифорки, на "1" (окупируемая позиция живого кораблика)

}

}

for (int g = 0; g < harakteristik[ship].paluba; g++)

{

for (int n = 0; n < 2; n++) //выполняем нижеуказаныый код 2 раза так как нам нужно поменять занчения в массиве для 2-ух клеточек (одной палубы)

{

korabl\_pos[y - 1 + g][x + n] = 13 - ship + plus; //меняем значение позиций в двумерном массиве который будем прверять на различные цифорки, на "4-13" (номера корабликов (у каждого свой))

}

}

ship++; //переходим к следующему кораблику

}

}

}

void RazrisovkaPos(int mas[12][24], int x, int num\_color) //прорисовка игровой карты в соответсвии с положением поставленных корабликов

{

for (size\_t i = 1; i < 11; i++) //выполнеям нижеуказанный код 10 раз так как общее количество строк равно 10

{

int y = 1 + i; //уставнавливаем координаты по умолчанию 2 по Х и 2 по Y так как отрисовка поля должно начинаться там

ustanovCursor(x, y); //перемещаем курсор на вышеуказанные позиции

for (size\_t g = 2; g < 22; g++) //выполнеям нижеуказанный код 20 раз так как общее количество столбцов равно 20

{

if (mas[i][g] == 2) //если в массиве на этом месте находится "2"

{

izmenColor(LightGray, LightGray); //рисуем серую клеточку

}

else if (mas[i][g] == 3) //если в массиве на этом месте находится "3"

{

izmenColor(LightMagenta, LightMagenta); //рисуем клеточку непонятного цвета

}

else if (mas[i][g] >= 4) //если в массиве на этом месте находится "4+"

{

izmenColor(Color(num\_color), Color(num\_color)); //устанавливаем цвет корабликов (наши - черные / врага - белые)

}

else //в иных случаях рисуем белую клеточку

{

izmenColor(White, White); //устанавливаем белый цвет так как таким цветом будет игровая локация на которой будут происходить основные действия

}

cout << "+"; //рисуем любой знак, из-за окраски он будет квадратиком

}

cout << "\n"; //переход на новую строку при отрисовки одной строки полностью

}

}

void Nachalo() //функция которая будет показана игроку самая первая

{

izmenColor(White, LightBlue); //Устанавливаем цвет текста (белый на голубом)

//SetCursor(24, 12); //устанавливаем позицию курсора (на своё усмотренние)

ustanovCursor(50, 20);

cout << "МОРСКОЙ БОЙ" << endl; //выводим надпись на экран

cout << " Для передвижения курсора \n"

" используем [клавиши управления курсором] \n"

" Для переворачивания корабля клавиша [Й] \n"

" для ввода [ввод] " << endl;

}

void SozdatPole(int skorost, int X, int num\_color) //Функция благодоря которой будет отрисовываться игровое поле

{

ustanovCursor(X, 1); //Устанавливаем курсор на нужную позицию

char mas[10] = { 'А','Б','В','Г','Д','Е','Ж','З','И','К' }; //буковки которые будут находится сверху игрового поля и ориентировать игрока куда он бьёт

izmenColor(White, Color(num\_color)); //Смена цвета (буква - белый; заливка - красный/синий)

int A = 0, B = 0; //переменные необходимые для правильной отрисовки игрового поля

cout << " " << mas[0]; //С помощью этого печатаем букву "А"и ставим в начале пробел чтобы было ровненько и красиво

for (size\_t i = 0; i < 12; i++) //выполнеям нижеуказанный код 12 раз так как общее количество строк равно 12

{

if (B == 0) //гениальная проврека, которая сработает один раз

{

ustanovCursor(X + 3, 1 + i); //Устанавливаем курсор на нужную позицию

B++; //увеличиваем значение переменной

}

else //иначе

{

ustanovCursor(X, 1 + i); //Устанавливаем курсор на нужную позицию

}

for (size\_t g = 0; g < 23; g++) //выполнеям нижеуказанный код 23 раз так как общее количество столбцов равно 23

{

if (i == 0 && g == 22) //выполняем ниэеуказанный код, если мы на первой строке, и 22 элементе

{

izmenColor(Color(num\_color), Color(num\_color)); //Меняем цвет текста и заливки на красный/синий (границы поля на которых нету цифр ил букв)

Sleep(skorost); //Задержка чтобы была анимация

cout << '+'; //рисуем любой знак, из-за окраски он будет квадратиком

}

if (i == 0 && g > 0 && g < 10) //выполняем ниэеуказанный код, если мы на первой строке, и в диапозоне с 0-го по 10-ый элемент

{

izmenColor(White, Color(num\_color)); //Смена цвета (буква - белый; заливка - красный/синий)

Sleep(skorost); //Задержка чтобы была анимация

cout << " " << mas[g]; //Пишем пробел, и букву которая стоит на месте элемента номер которого равен значению "g"

}

else if (i > 0 && i < 11 && g == 0) //Иначе если мы находимся с 0-ой по 11-ую строку на 0-м элементе

{

izmenColor(White, Color(num\_color)); //Смена цвета (буква - белый; заливка - красный/синий)

if (i < 10) //Если мы находимся на строках не достигающих значения "10"

{

Sleep(skorost); //Задержка чтобы была анимация

cout << i << ' '; //Пишим цифру указывающая на номер строки которая будет помагать игроку ориентироваться куда он стреляет () для того чтобы было ровно по соотношению с двузначными цифрами

}

else //иначе

{

Sleep(skorost); //Задержка чтобы была анимация

cout << i; //Пишем цифру без пробелов так как тут они не будут нужны ведь цифра которая будет выводиться "10"

}

}

else if (i == 11 || g > 20 && g < 23) //Иначе если мы находимся на 11-ой строке, на 21-22-ых элементах

{

if (A == 0 && i == 11) //проверка на последней строке которая нужна чтобы выводился кубик который не выводится самостоятельно

{

Sleep(skorost); //Задержка чтобы была анимация

cout << "+";//рисуем любой знак, из-за окраски он будет квадратиком

A++; //меняем значение переменной благодаря которой было можно реализовать дорисовку нужного кубика

}

izmenColor(Color(num\_color), Color(num\_color)); //Меняем цвет текста и заливки на красный/синий (границы поля на которых нету цифр ил букв)

Sleep(skorost); //Задержка чтобы была анимация

cout << '+'; //рисуем любой знак, из-за окраски он будет квадратиком

}

else if (i > 0) //если мы находимся на строке номер которой не равен "0" (непосредственно игровое поле)

{

Sleep(skorost); //Задержка чтобы была анимация

izmenColor(White, White); //Меняем цвет текста и заливки на белый (непосредственно игровое поле)

cout << '+'; //рисуем любой знак, из-за окраски он будет квадратиком

}

}

}

izmenColor(Black, Black); //устанавливаем полностью черный цвет

}

void MenuVyboraRastanovki(char variant[2][30], int T) //меню выбора расстановки

{

system("cls"); //очищаем консоль

for (size\_t i = 0; i < 2; i++) //создаём цикл для 2-ух элементов

{

ustanovCursor(30, 6 + i); //Устанавливаем курсор на нужную позицию

for (size\_t p = 0; p < 30; p++) //создаём цикл для всех 30-ти буковок

{

if (i == T) //проверка на выбранный пункт

{

izmenColor(Black, White); //выделяем выбранный пункт

}

else //иначе

{

izmenColor(White, Black); //убираем выделение выбранного пункта

}

cout << variant[i][p]; //выводим пункты в виде тектса

}

}

SozdatPole(0, 0, 9); //создаём игровое поле занаво

}

void Repliki(int num)//реплики

{

char variant[11][40] = { "ВРАГ ДУМАЕТ КУДА УДАРИТЬ...","ВРАГ ПРОМАЗАЛ!", "ВРАГ ПОПАЛ!", "ВРАГ ПОБЕДИЛ... НЕ ГРУСТИ...", "ТВОЙ ХОД." , "ТЫ ПРОМАЗАЛ...", "ТЫ ПОПАЛ!", "ТЫ ПОБЕДИЛ, ПОЗДРАВЛЯЮ!", "+++++++++++++++++++++++++++++++++++++++", "ТЫ ПОПАЛ, ХОДИ СНОВА!", "ЮХУ, ТЫ УНИЧТОЖИЛ ВРАЖЕСКИЙ КОРАБЛИК!" };

//список вариантов реплик которые могут вывестись во время игры

ustanovCursor(22, 15); //Устанавливаем курсор на нужную позицию

izmenColor(Black, Black); //устанавливаем полностью черный цвет

for (size\_t i = 0; i < 25; i++) //цикл для стирание предыдущего текста

{

cout << variant[10][i]; //выводим "+"

cout << variant[10][i]; //выводим "+"

}

ustanovCursor(22, 15); //Устанавливаем курсор на нужную позицию

if (num > 0 && num < 4) //проверяем, если реплики больше относятся к врагу

{

izmenColor(Red, Black); //Устанавливаем красный цвет текста на черном фоне

}

else //иначе реплики больше относятся ко мне

{

izmenColor(Green, Black); //Устанавливаем зеленый цвет текста на черном фоне

}

for (size\_t i = 0; i < 30; i++) //создаём цикл для всех 30-ти буковок

{

cout << variant[num][i]; //выводим реплики в виде тектса

}

}