****

**КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО**

**ОБРАЗОВАНИЯ**

|  |
| --- |
|  |
|  |
|  |
|  |

**Отчет**

**Специальность 09.02.02 «Компьютерные сети»**

**Основы объектно-ориентированного программирования**

Выполнила студентка группы 32КС-20

Мусаткина А.С.

Проверила

Кукшева Б.А.

**Москва 2022**

**Оглавление**

[**Введение** 3](#_Toc116834070)

[**1.** **Создание игрового поля.** 4](#_Toc116834071)

[**2.** **Добавление функции стрельбы.** 12](#_Toc116834072)

[**3.** **Создание бота.** 15](#_Toc116834073)

[**Заключение** 21](#_Toc116834074)

[**Список источников** 22](#_Toc116834075)

# **Введение**

В данной работе передо мной была поставлена цель *создать любую игру на выбор на языке C# в программе Visual Studio.* Из предложенных вариантов меня заинтересовала тема «Морской бой», поскольку на мой взгляд эта игра не очень сложная, но при этом очень увлекательна.

Для достижения цели были поставлены следующие задачи:

* Изучить структуру игры;
* При помощи видеоматериала изучить команды.

**Правила игры морской бой.**

Начало игры

В игре принимают участие два игрока. Они берут по одному игровому набору, располагаются так, чтобы не видеть игровое поле противника, и готовятся к игре. На игровых полях располагаются корабли.

Внимание! Корабли не должны соприкасаться, то есть расстояние между ними должно быть не менее одной клетки.

Внимание! Переставлять свои корабли после начала игры категорически запрещено.

**Ход игры**

После того как корабли расставлены, можно начинать игру.

Существуют два основных варианта игры:

1. Стрельбу игроки ведут строго по очереди.

2. Стрельбу игроки ведут "до первого промаха", то есть если игрок попал в корабль противника, он производит следующий выстрел, и только после его промаха ход переходит к другому игроку.

На игровом поле размещают свои корабли и отражают результаты выстрелов противника. Экран служит, чтобы отражать результаты собственных выстрелов по противнику. На поле и на экране промах обозначается черным цветом, а попадание — синим. Каждый выстрел имеет свои координаты. Стреляющий должен называть координаты громко и четко. Например: «Выстрел по А4!»

Игрок, по кораблям которого ведется огонь, отмечает выстрел на своем игровом поле:

* в случае промаха - черным
* в случае попадания - синим

Затем он четко объявляет результат выстрела:

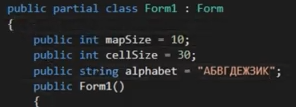
* вариант 1. "Промах!" - игрок не попал в корабль
* вариант 2. "Ранил!" - игрок попал в корабль, но не уничтожил.
* вариант 3. "Потопил!" - корабль уничтожен.

Узнав результат своего выстрела, игрок отмечает его черным или синим на экране. Чтобы уничтожить корабль, требуется определенное количество попаданий. Оно соответствует числе отверстий под синим фишки на его палубе.

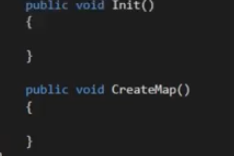
Победитель тот, кто первым уничтожит все корабли противника.

# **Создание игрового поля.**

После создания Form1 объявляем необходимых переменные: размер карты, размер ячейки и за инциденты поля.



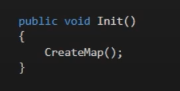
Добавляем функции Init и CreateMap.



Создаем два массива первый для карты игрока, второй для карты бота.



В функции Init мы вызовем создание нашей карты.

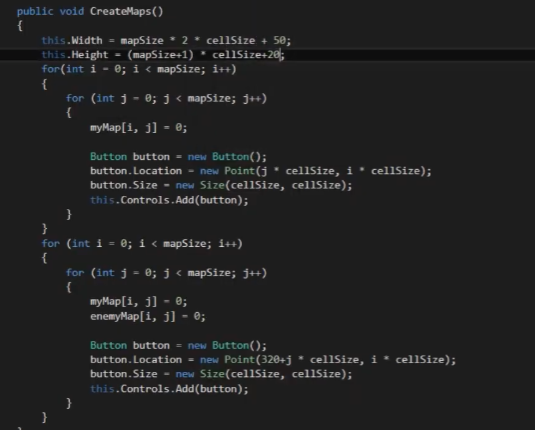


В функции CreateMap прописано наше игровое поле. Разбиваем на два цикла: первый цикл – карта игрока, второй – карта бота.

В цикле заполняем массив полями и создаем кнопку с определенными координатами, которые высчитываем через переменные цикла.

В начале этой функции рассчитаем ширину и высоту всей формы, чтобы он автоматически подстраивался под нужный размер карты.

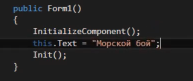
Во втором цикле сделали тоже самое, но сделали отдаление на 320, чтобы наша карта была правее.



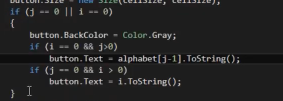
По итогу, запустив проект мы должны получить следующее:



В Form1 зададим название нашей формы «Морской бой».

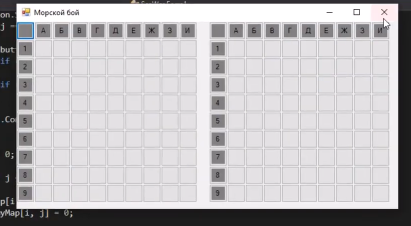


Теперь дополним нашу функцию создания карт – окрашиваем первую строку и столб в серый цвет для того, чтобы оставить эти ячейки для позиций индексов и добавляем условие, при котором индексы не будут начинаться с первой ячейки.

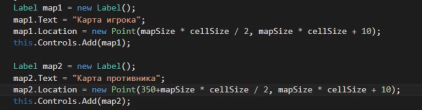


Скопируем этот цикл и ставим в цикл для карты бота.

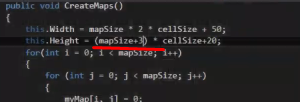
В итоге можем наблюдать, что для обоих карт были созданы правильные распределения индексов.



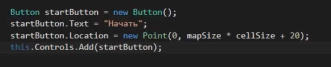
Создаем новый label, чтобы обозначить, где находится карта игрока, а где карта противника.



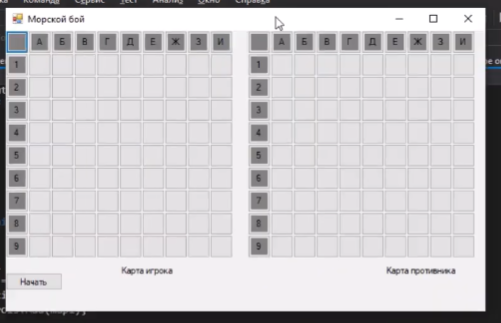
В функции CreateMap увеличим размер карты, чтобы в будущем добавить кнопку.



Теперь прописываем код для создания кнопки старта.

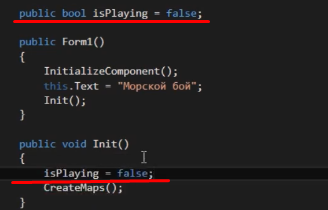


Таким образом, запустив проект, мы должны получить такое игровое поле:

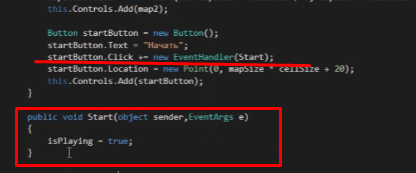


Создаем новую переменную isPlaying, которая будет отображать нажал ли игрок на кнопку.

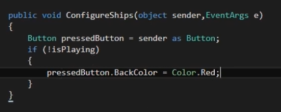
В функции Init присваиваем этой переменной false.



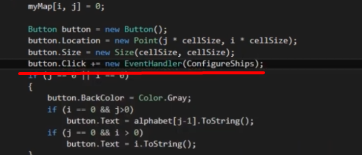
Теперь добавим функцию, которая будет выполнять роль обработчика нажатия кнопки. В ней будет просто будет присвоено значение true, а саму обработку добавим в то место, где была создана эта кнопка.



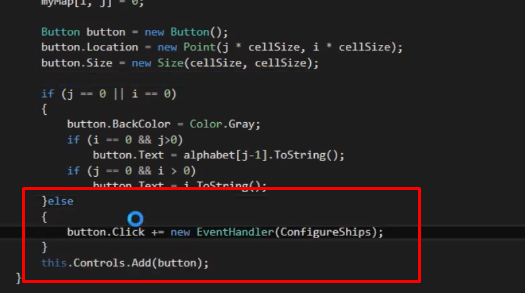
Добавим функцию расстановки кораблей. В начале мы прописываем код, который проверяет нажал ли игрок на кнопку. Добавляем цикл, при котором если игрок нажал на кнопку, то все ячейки будут окрашиваться в красный при нажатии.



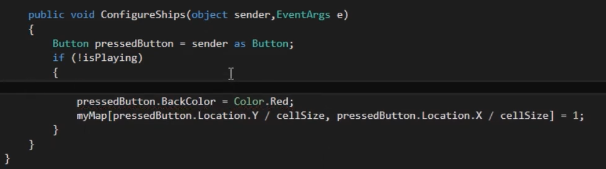
Однако в функции CreateMap нам нужно добавить событие и привязать это событие к нашим кнопкам.



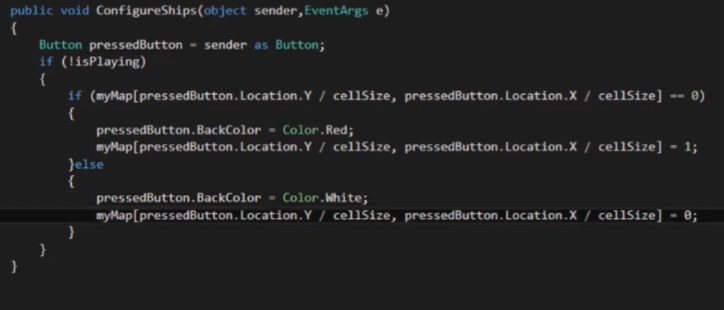
Теперь допишем else, чтобы мы не могли окрашивать в красный первую строку и столб, где указаны индексы.



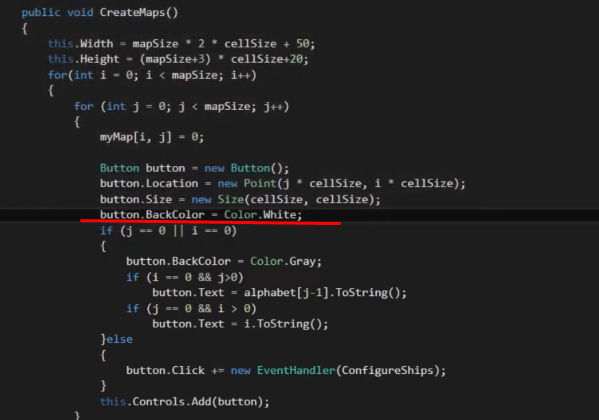
Добавим в само условие раскраски, чтобы мы не могли красить ячейки, на которых есть буквы и цифры.

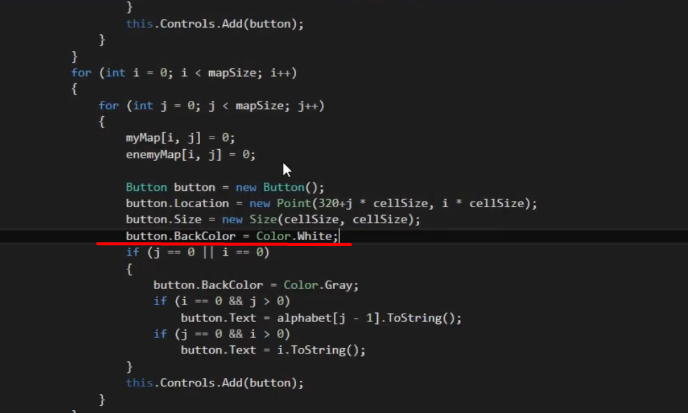


Также мы должны проверить, если ячейка нулевая, то мы заполняем ее красным, если там уже что-то есть, то мы ее очищаем.

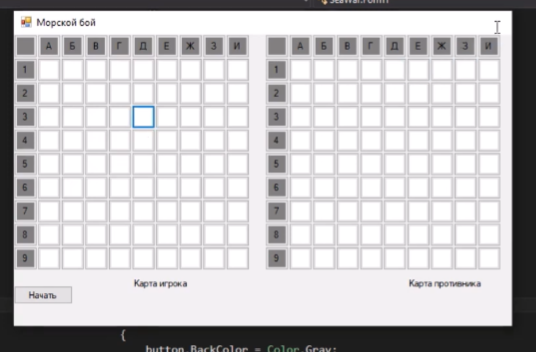


Теперь в циклах формах добавим изменение цвета на белый наших ячеек на карте игрока и противника.





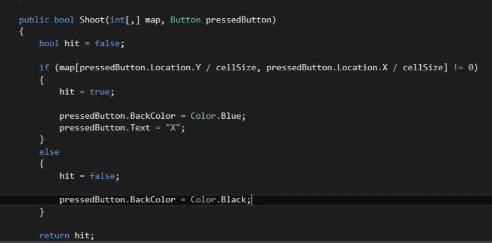
По итогу наше поле теперь должно выглядеть вот так:



# **Добавление функции стрельбы.**

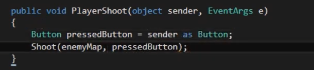
Теперь добавим функцию Shoot, с помощью которой будет осуществляться стрельба по полю противника. В эту функцию мы придадим нажатую кнопку.

Сначала проверяем есть ли у нас в координатах нажатые кнопки относительно нашей карты и присвоим значение 0. Если что-то есть, то переменная hit = true и красим эту кнопку в голубой цвет, если не попали, то hit= false, то мы красим эту кнопку в черный цвет.

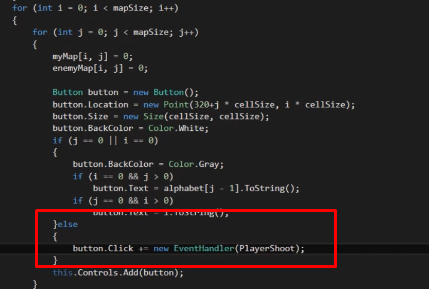


Теперь напишем функцию PlayerShoot. Эта функция будет выполнять роль обработчика событий нажатия кнопки.

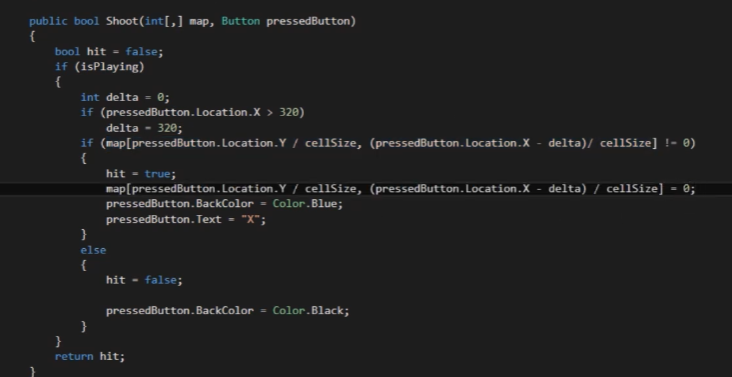
Здесь мы будем вызывать функцию Shoot и передавать туда карту врага и нажатую кнопку.



Теперь нам нужно повесить этот обработчик на кнопки карты врага. В цикле else присваиваем обработчик.



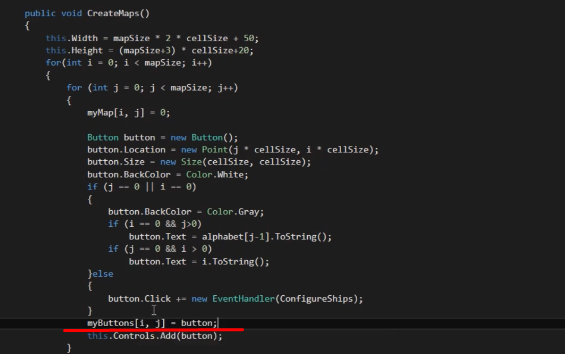
Также не забываем, что у нас было смещение и это нужно учесть в функции Shoot. Для этого мы приведем переменную delta и присваиваем ей ноль. И смотрим, если позиция нажатой кнопки выше 320, то мы присваиваем смешению 320. И высчитываем это delta из Location X.

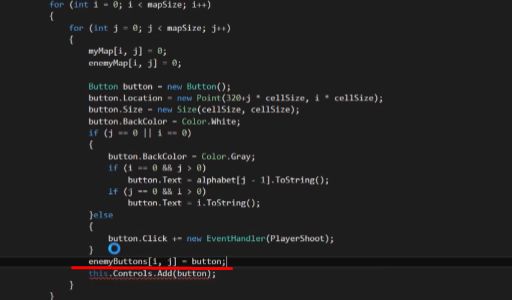


Теперь добавляем массив, в котором будем хранить кнопки игрока и кнопки противника.



И соответственно в этих двух циклах for кнопок добавим создание кнопок в двумерный массив.

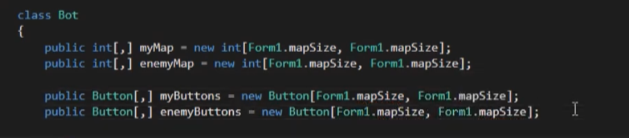




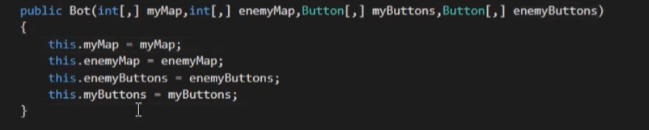
# **Создание бота.**

Теперь добавим новый скрипт, который назовем Bot.

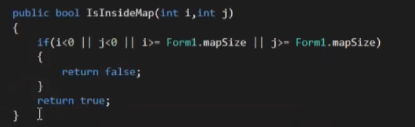
Копируем строчки из Form1, добавляем в класс bot и импортируем.



Теперь напишем конструктор для нашего бота. Здесь мы будем передавать его поле и поле игрока.



Напишем функцию IsInsideMap, который будет принимать два параметра, если какой-то из них будет выходить за границы нашей карты то примет значение false, в ином случае true.

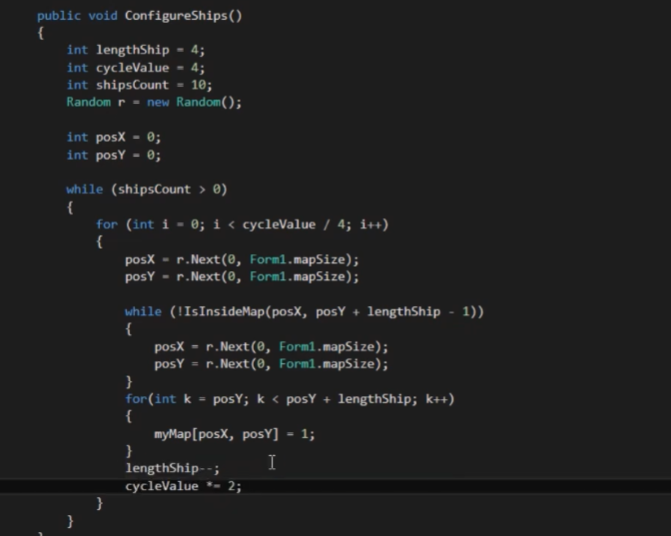


Добавим функцию, которая будет генерировать наши корабли для бота при помощи цикла while. Сначала мы добавляем количество кораблей и случайное генерирование.

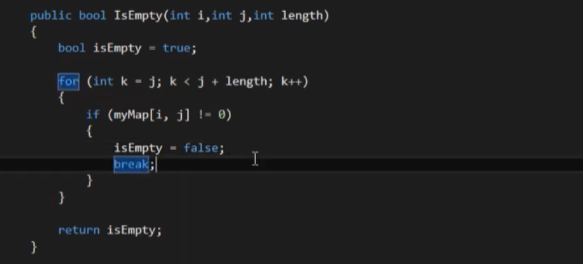
В цикле for через цикл while проверяем если корабли выходят за границу карты, то мы генерируем новый.

Теперь после ограничений выставляем корабли через еще один цикл for. Как раз той длины, которая будет на текущий момент операции этого алгоритма.

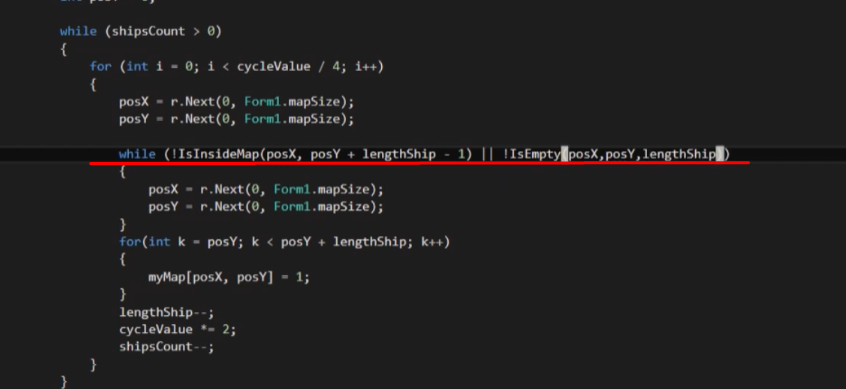
В конце мы уменьшаем длину корабля, так как у нас дальше корабль начинается от 4 доходит до 1.

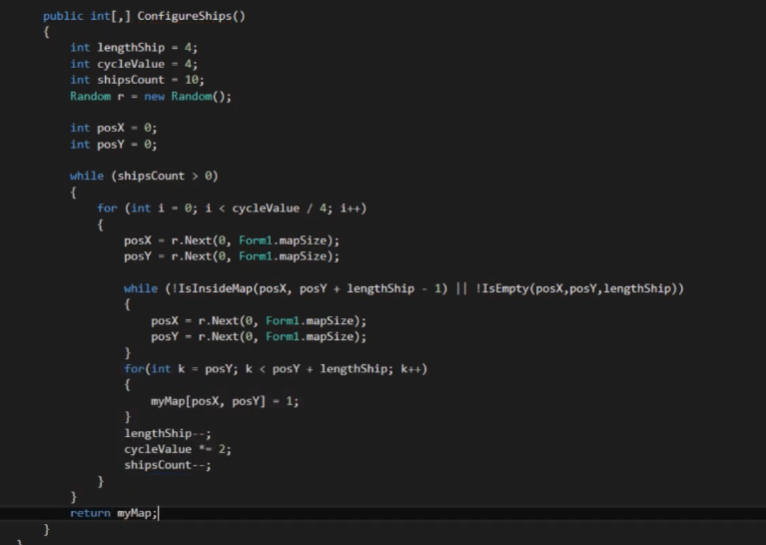


Теперь напишем функцию isEmpty, которая будет возвращать переменные по типу bool. Через нее также будем передавать длину корабля. И соответственно проверять, если этот параметр не нулевой в карте бота, то isEmpty = false.

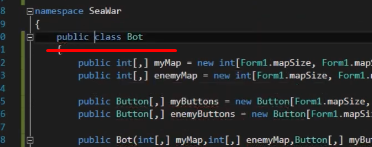


И этот проверку добавим в while, который проверяет не выходят ли позиции Y и X за границы карты.



Также изменим нашу функцию, чтобы она возвращала двумерный массив.

Теперь делаем этот класс публичным.



В основном скрипте мы пропишем нашего бота и в конструкторе формы (функции Init) создадим экземпляр бота. Затем в карте противника запишем возвращенную карту с конфигурацией кораблей.

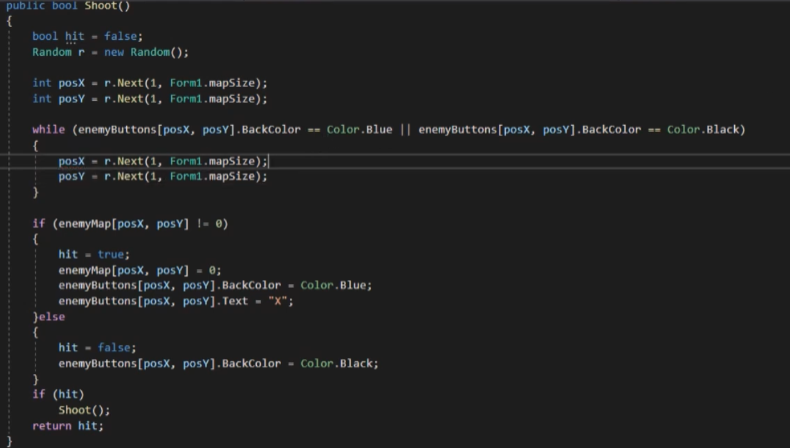


Скопируем функцию shoot и напишем его для бота. Здесь нам также понадобится генерация случайных чисел, и posX и posY, в которые будем записывать случайное число.

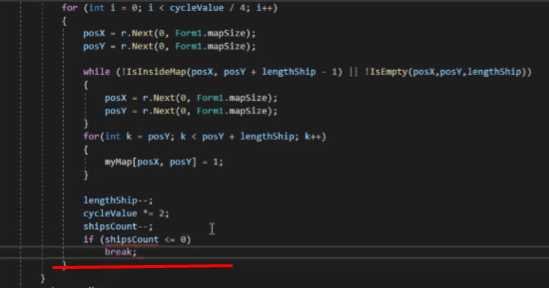
Пишем проверку, если у нас в этих координатах что-то есть, то значит hit = true и эту ячейку закрасить в синий цвет, то есть мы попали в корабль. В ином случае кнопка краситься в черный и hit = false.

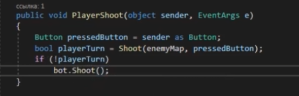
Также добавим, что при попадании бот ходит еще раз.

Не забудем дописать проверку, которая будет проверять, если цвет ячейки синий или черный, то это значит, что туда уже стреляли.



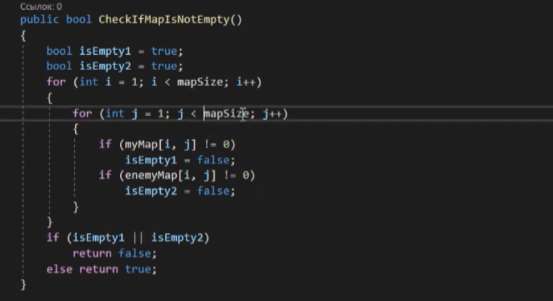
Также дополним цикл for в генерации кораблей, если мы это не добавим, то у нас будет неправильно работать алгоритм.



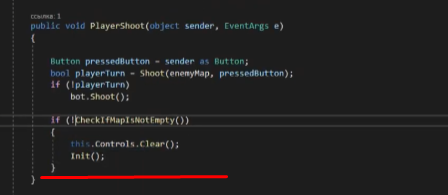


Теперь напишем функцию для основного скрипта, которая будет проверять есть ли на карте еще живой корабли для карты игрока и бота. Соответственно, если нету, то будем возвращать false, если есть true.

И в конце добавим, если карта у нас пустая, то вернем false. Иначе вернем true.



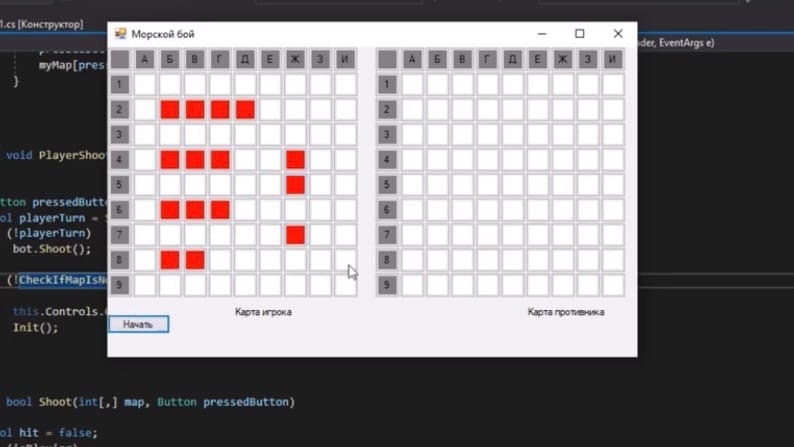
В обработчике PlayerShoot добавим, когда закончится стрельба для всех то игра начинается заново.



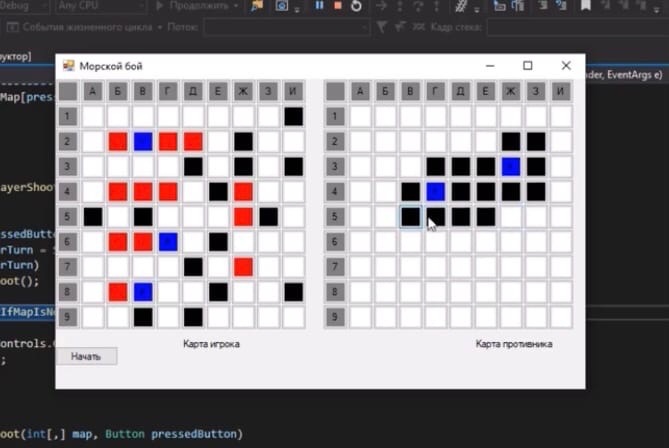
# **Заключение**

Передо мной стояла задача создания рабочей игры морской бой, и как я считаю я выполнила данное задание. Игра получилась не слишком замысловатой и понятной для любого пользователя. Также, мне удалось сделать подробную инструкцию по созданию самой игры, которую может создать любой человек без особых проблем.

Сначала мы расставляем наши корабли в любом расположении и нажимаем на кнопку "начать". После кнопки начать мы делаем ход, в случае попадания делаем еще один ход, а в противном случае начинает ходить бот.



Игра будет продолжаться, пока кто-то не найдет все корабли противника.



**Программный код для Form1.**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SeaWars

{

public partial class Form1 : Form

{

public const int mapSize = 10;

public int cellSize = 30;

public string alphabet = "АБВГДЕЖЗИК";

public int[,] myMap = new int[mapSize, mapSize];

public int[,] enemyMap = new int[mapSize, mapSize];

public Button[,] myButtons = new Button[mapSize, mapSize];

public Button[,] enemyButtons = new Button[mapSize, mapSize];

public bool isPlaying = false;

public Bot bot;

public Form1()

{

InitializeComponent();

this.Text = "Морской бой";

Init();

}

public void Init()

{

isPlaying = false;

CreateMaps();

bot = new Bot(enemyMap, myMap, enemyButtons, myButtons);

enemyMap = bot.ConfigureShips();

}

public void CreateMaps()

{

this.Width = mapSize \* 2 \* cellSize + 50;

this.Height = (mapSize + 3) \* cellSize + 20;

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

{

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

{

myMap[i, j] = 0;

Button button = new Button();

button.Location = new Point(j \* cellSize, i \* cellSize);

button.Size = new Size(cellSize, cellSize);

button.BackColor = Color.White;

if (j == 0 || i == 0)

{

button.BackColor = Color.Gray;

if (i == 0 && j > 0)

button.Text = alphabet[j - 1].ToString();

if (j == 0 && i > 0)

button.Text = i.ToString();

}

else

{

button.Click += new EventHandler(ConfigureShips);

}

myButtons[i, j] = button;

this.Controls.Add(button);

}

}

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

{

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

{

myMap[i, j] = 0;

enemyMap[i, j] = 0;

Button button = new Button();

button.Location = new Point(320 + j \* cellSize, i \* cellSize);

button.Size = new Size(cellSize, cellSize);

button.BackColor = Color.White;

if (j == 0 || i == 0)

{

button.BackColor = Color.Gray;

if (i == 0 && j > 0)

button.Text = alphabet[j - 1].ToString();

if (j == 0 && i > 0)

button.Text = i.ToString();

}

else

{

button.Click += new EventHandler(PlayerShoot);

}

enemyButtons[i, j] = button;

this.Controls.Add(button);

}

}

Label map1 = new Label();

map1.Text = "Карта игрока";

map1.Location = new Point(mapSize \* cellSize / 2, mapSize \* cellSize + 10);

this.Controls.Add(map1);

Label map2 = new Label();

map2.Text = "Карта противника";

map2.Location = new Point(350 + mapSize \* cellSize / 2, mapSize \* cellSize + 10);

this.Controls.Add(map2);

Button startButton = new Button();

startButton.Text = "Начать";

startButton.Click += new EventHandler(Start);

startButton.Location = new Point(0, mapSize \* cellSize + 20);

this.Controls.Add(startButton);

}

public void Start(object sender, EventArgs e)

{

isPlaying = true;

}

public bool CheckIfMapIsNotEmpty()

{

bool isEmpty1 = true;

bool isEmpty2 = true;

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

{

for (int j = 1; j < mapSize; j++)

{

if (myMap[i, j] != 0)

isEmpty1 = false;

if (enemyMap[i, j] != 0)

isEmpty2 = false;

}

}

if (isEmpty1 || isEmpty2)

return false;

else return true;

}

public void ConfigureShips(object sender, EventArgs e)

{

Button pressedButton = sender as Button;

if (!isPlaying)

{

if (myMap[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] == 0)

{

pressedButton.BackColor = Color.Red;

myMap[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] = 1;

}

else

{

pressedButton.BackColor = Color.White;

myMap[pressedButton.Location.Y / cellSize, pressedButton.Location.X / cellSize] = 0;

}

}

}

public void PlayerShoot(object sender, EventArgs e)

{

Button pressedButton = sender as Button;

bool playerTurn = Shoot(enemyMap, pressedButton);

if (!playerTurn)

bot.Shoot();

if (!CheckIfMapIsNotEmpty())

{

this.Controls.Clear();

Init();

}

}

public bool Shoot(int[,] map, Button pressedButton)

{

bool hit = false;

if (isPlaying)

{

int delta = 0;

if (pressedButton.Location.X > 320)

delta = 320;

if (map[pressedButton.Location.Y / cellSize, (pressedButton.Location.X - delta) / cellSize] != 0)

{

hit = true;

map[pressedButton.Location.Y / cellSize, (pressedButton.Location.X - delta) / cellSize] = 0;

pressedButton.BackColor = Color.Blue;

pressedButton.Text = "X";

}

else

{

hit = false;

pressedButton.BackColor = Color.Black;

}

}

return hit;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

**Программный код для класса бот.**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace SeaWars

{

public class Bot

{

public int[,] myMap = new int[Form1.mapSize, Form1.mapSize];

public int[,] enemyMap = new int[Form1.mapSize, Form1.mapSize];

public Button[,] myButtons = new Button[Form1.mapSize, Form1.mapSize];

public Button[,] enemyButtons = new Button[Form1.mapSize, Form1.mapSize];

public Bot(int[,] myMap,int[,] enemyMap,Button[,] myButtons,Button[,] enemyButtons)

{

this.myMap = myMap;

this.enemyMap = enemyMap;

this.enemyButtons = enemyButtons;

this.myButtons = myButtons;

}

public bool IsInsideMap(int i,int j)

{

if(i<0 || j<0 || i>= Form1.mapSize || j>= Form1.mapSize)

{

return false;

}

return true;

}

public bool IsEmpty(int i,int j,int length)

{

bool isEmpty = true;

for (int k = j; k < j + length; k++)

{

if (myMap[i, k] != 0)

{

isEmpty = false;

break;

}

}

return isEmpty;

}

public int[,] ConfigureShips()

{

int lengthShip = 4;

int cycleValue = 4;

int shipsCount = 10;

Random r = new Random();

int posX = 0;

int posY = 0;

while (shipsCount > 0)

{

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

{

posX = r.Next(1, Form1.mapSize);

posY = r.Next(1, Form1.mapSize);

while (!IsInsideMap(posX, posY + lengthShip - 1) || !IsEmpty(posX,posY,lengthShip))

{

posX = r.Next(1, Form1.mapSize);

posY = r.Next(1, Form1.mapSize);

}

for(int k = posY; k < posY + lengthShip; k++)

{

myMap[posX, k] = 1;

}

shipsCount--;

if (shipsCount <= 0)

break;

}

cycleValue += 4;

lengthShip--;

}

return myMap;

}

public bool Shoot()

{

bool hit = false;

Random r = new Random();

int posX = r.Next(1, Form1.mapSize);

int posY = r.Next(1, Form1.mapSize);

while (enemyButtons[posX, posY].BackColor == Color.Blue || enemyButtons[posX, posY].BackColor == Color.Black)

{

posX = r.Next(1, Form1.mapSize);

posY = r.Next(1, Form1.mapSize);

}

if (enemyMap[posX, posY] != 0)

{

hit = true;

enemyMap[posX, posY] = 0;

enemyButtons[posX, posY].BackColor = Color.Blue;

enemyButtons[posX, posY].Text = "X";

}else

{

hit = false;

enemyButtons[posX, posY].BackColor = Color.Black;

}

if (hit)

Shoot();

return hit;

}

}

}

# **Список источников**

<https://www.youtube.com/watch?v=LhLpLOZFE74>