|  |  |
| --- | --- |
|  | logo_kmpo  **КОЛЛЕДЖ**  **МНОГОУРОВНЕВОГО**  **ПРОФЕССИОНАЛЬНОГО**  **ОБРАЗОВАНИЯ** |

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА И ГОСУДАРСТВЕННОЙ СЛУЖБЫ ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

КОЛЛЕДЖ МНОГОУРОВНЕВОГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

**ОТЧЁТ**

**По дисциплине:** Основы объектно-ориентированного программирования.

Выполнила студентка

3 курса

Группы 31КС-20

Колосова Василиса

Специальность 09.02.02

«Компьютерные сети»

Преподаватель

Кукшева Б.А

Москва 2022.

**Введение.**

Для того чтобы решить поставленную задачу нужно осознавать, о чем идет речь и понимать суть вопроса. Для этого необходимо знать основные понятия о программировании.

Программированием называется процесс создания компьютерных программ. Компьютерная программа является связкой многочисленных строк специального текста. Он является специальным, потому что создан таким образом, чтобы машине было понятно, какие действия должны быть выполнены.

Компьютерный код — это специальный текст, состоящий из набора пошаговых инструкций. Он не всегда содержит в себе нули и единицы, также в нём есть определённые слова и дополнительные символы. Компьютер считывает код, который сообщает ему, какие операции следует выполнить с данными. Для создания программ программист, должен уметь правильно давать команды компьютеру, для этого были придуманы специальные языки- языки программирования.

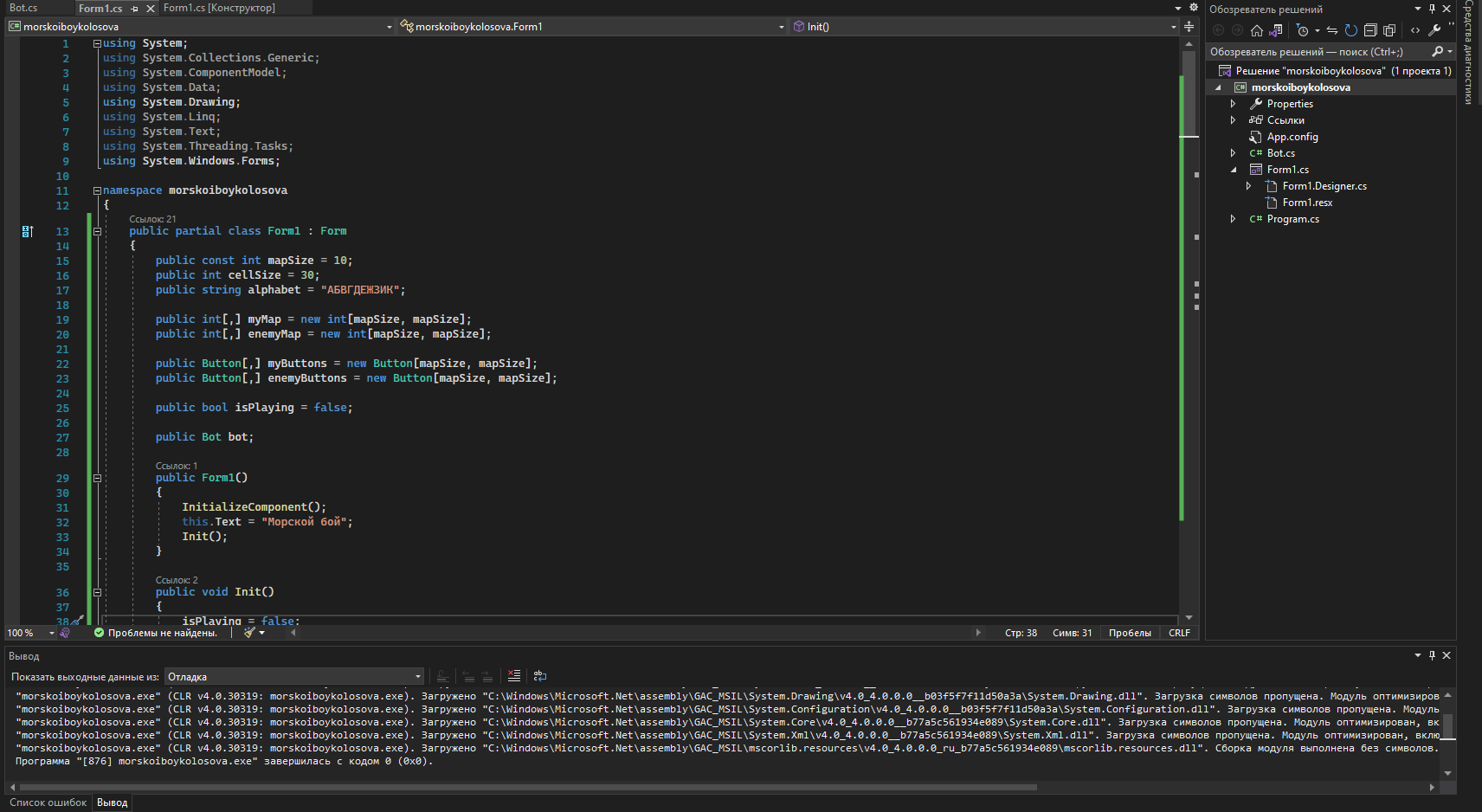
Язык программирования — формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программ и действия, которые выполнит исполнитель (компьютер) под её управлением. Получается, что для разных задач нужен определенный язык программирования.

Передо мной стоит задача создания программ для компьютерных систем и интерфейса для удобной работы пользователя с программой. Для выполнения я решил использовать язык С#, так как он прост в освоении и позволяет гибко изменять интерфейс.

Мое задание заключается в создании игры-бота «Морской бой».

**Разработка программного продукта.**

Для с программой нужно создать одну форму и класс. Form1 прописываю переменные, а в классе Bot.cs описала переменную Bot.

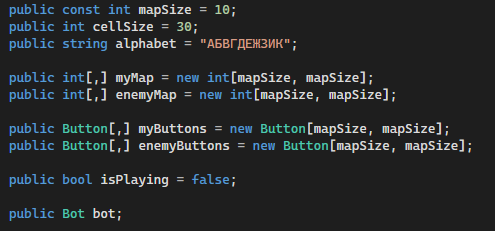


**Рисунок 1. Форма и класс.**

Объявляю размер карты и размер ячейки. И также делаю строку для называемых объектов, куда закидываю буквы для создания карты, чтоб мы могли прописать как в «Морском бое» ячейки. По горизонтали будут буквы, а по вертикали – цифры.

Прописала массивы, где myMap - моя карта, enemyMap – карта противника.

Public bool isPlaying – переменная, которая будет отоборажать что происходит. Нажал ли игрок на кнопку.

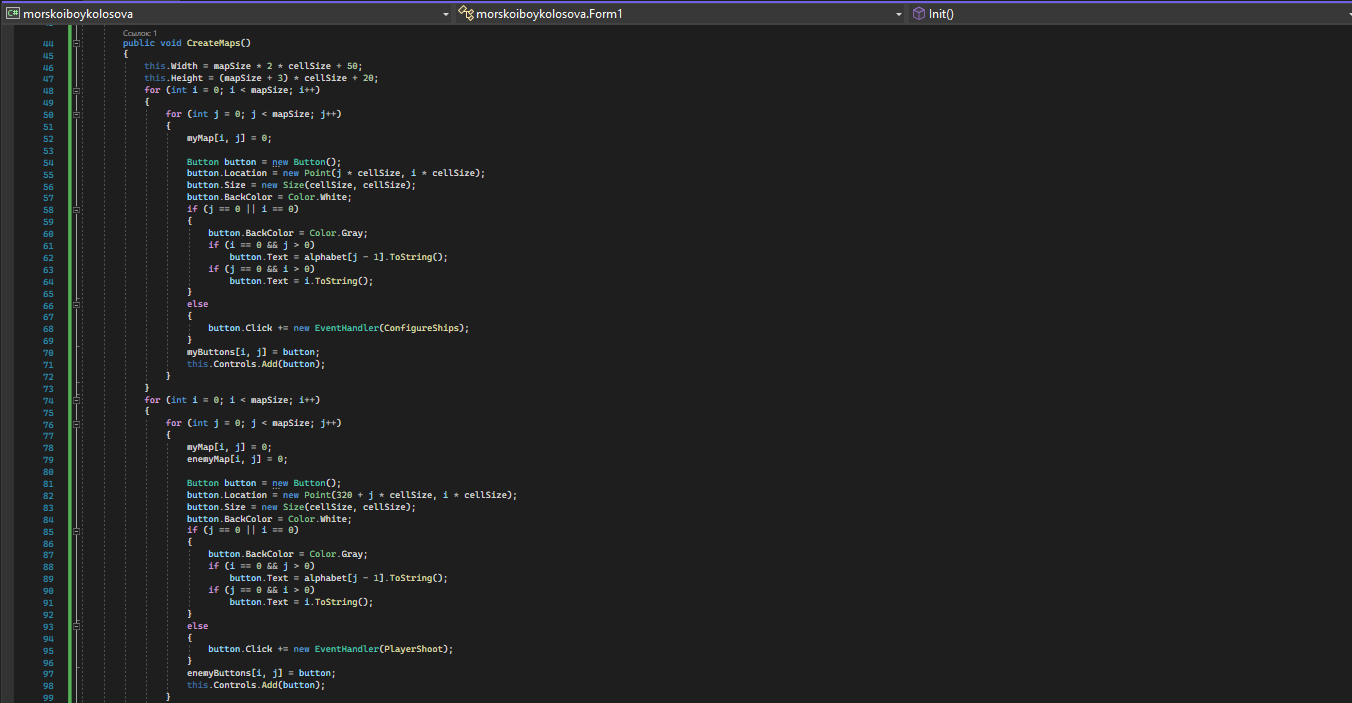


**Рисунок 2.Формат полей, строка букв**

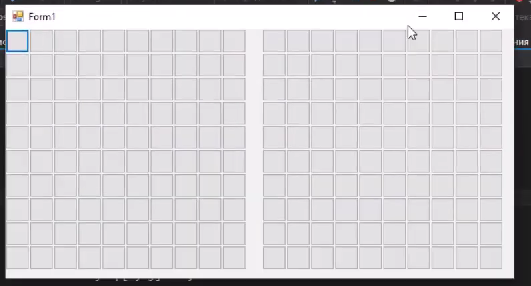
Дальше прописываю функции создания наших карт, игрового поля, используя двумерные массивы. В первом цикле идёт конфигурация карты игрока, во втором цикле идёт конфигурация карты противника/бота.

Создаём кнопку с определёнными координатами, которые мы высчитываем через переменные циклы и переменные, которые уже у нас были. Также в начале рассчитали ширину и высоту всей формы, чтоб она автоматически подгонялась под размер карты.

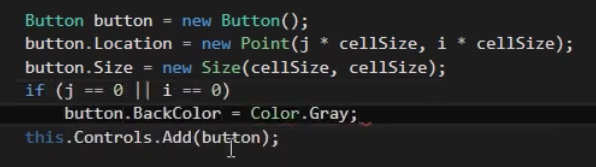
Во втором цикле мы тоже сделали автозаполнение, но со смещением.



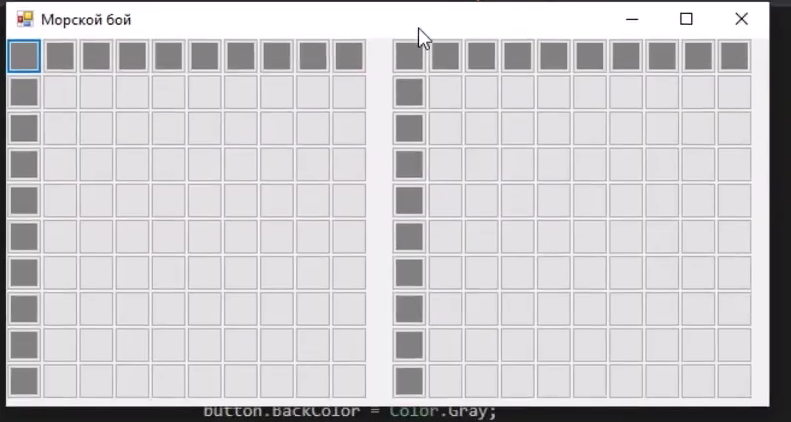
**Рисунок 3. код. Мой проект.**



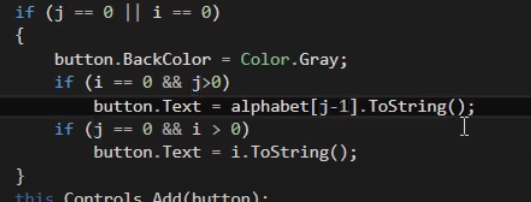
**Рисунок 4. Результат формы.**



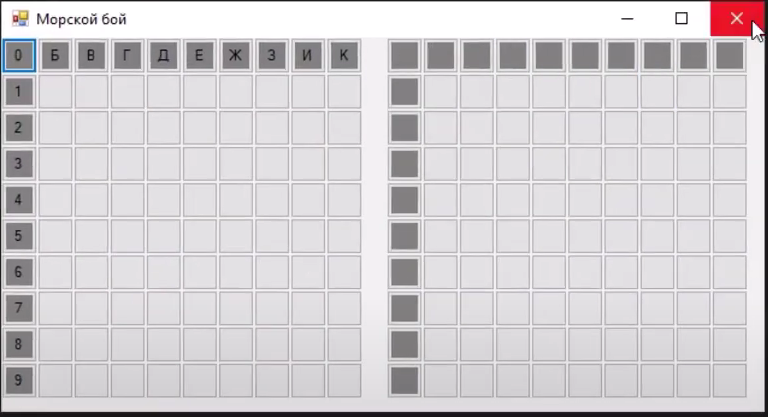
**Рисунок 5. Задаём цвет кнопкам.**



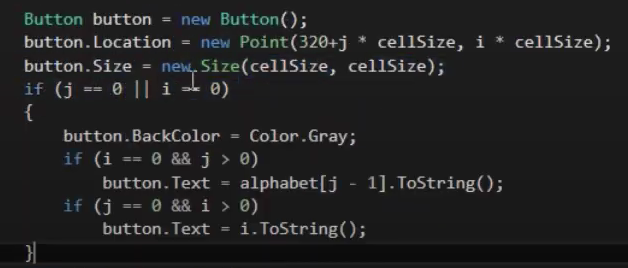
**Рисунок 6. Результат.**



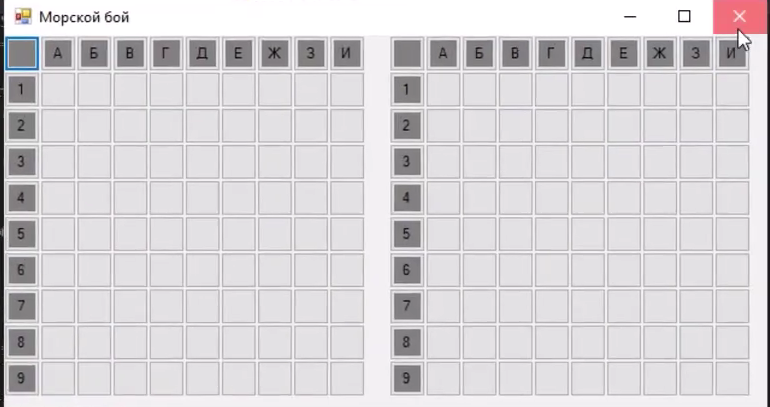
**Рисунок 7. Если i = 0 && j>0, то вводится кусочек текста(буквы). Если j = 0 && i>0, то вводятся цифры.**



**Рисунок 8. Результат.**

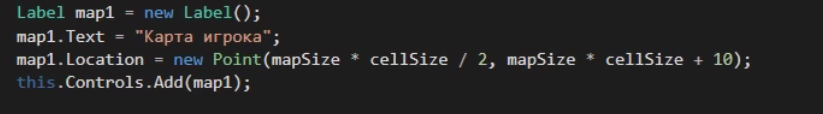


**Рисунок 9. Вводим тоже самое в наш второй цикл.**

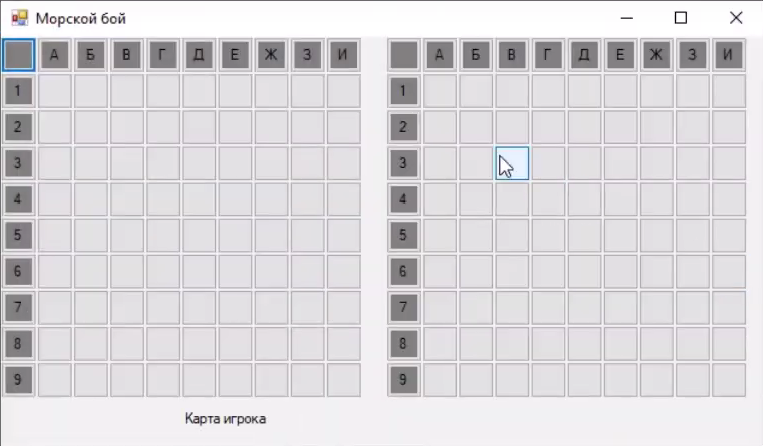


**Рисунок 10. Результат.**

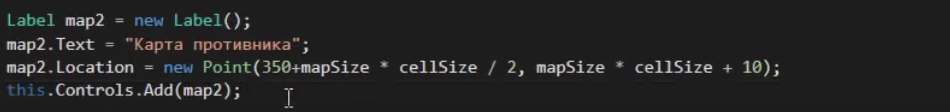
С помощью Label обозначим, чтоб было понятно, где находится карта противника и игрока. Позицию мы высчитываем исходя из наших значений, которые мы имеем. После всего это добавляем лейбл на нашу форму.



**Рисунок 11. код.**

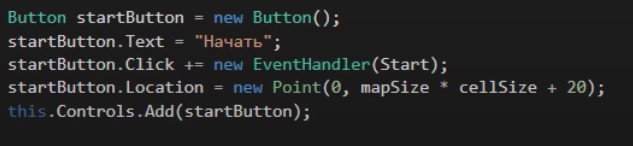


**Рисунок 12. Результат.**

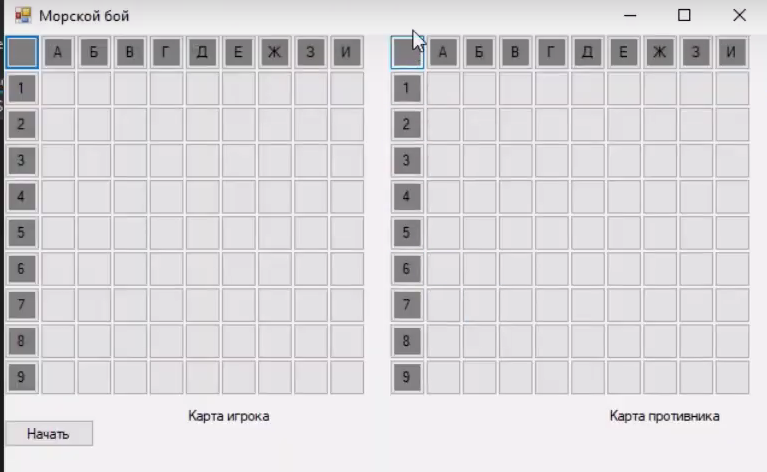


**Рисунок 13. создаём тоже самое для карты противника.**

Создаём кнопку для старта игры. Задаём ей текст и позицию и добавляем её на нашу форму.

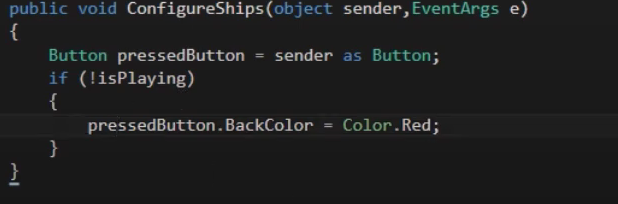


**Рисунок 14. создание кнопки "Начать"**

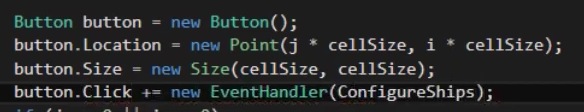


**Рисунок 15. Результат. Кнопка отобразилась.**

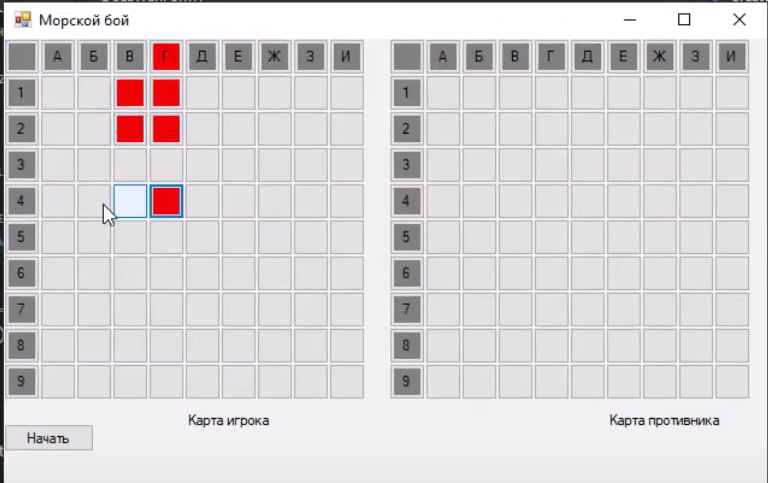
Теперь прописываем функцию расстановки наших кораблей на поле.



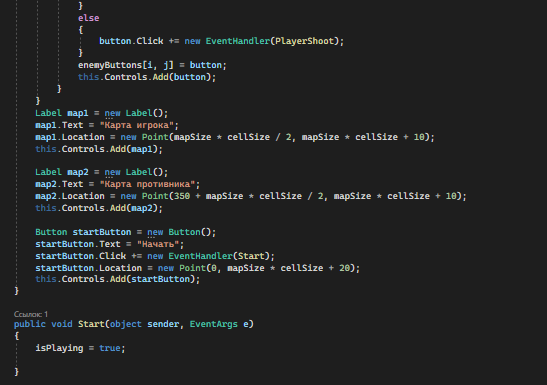
**Рисунок 16. Код. Здесь мы можем проверить. Если у нас не игровой режим, то мы можем конфигурировать наши корабли.**



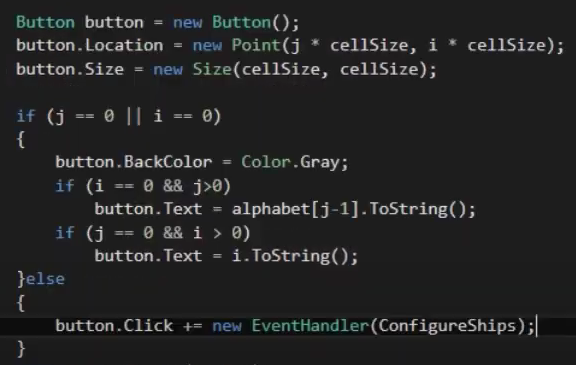
**Рисунок 17. В начале в переменную кнопки пропишем кнопку и привяжем событие.**



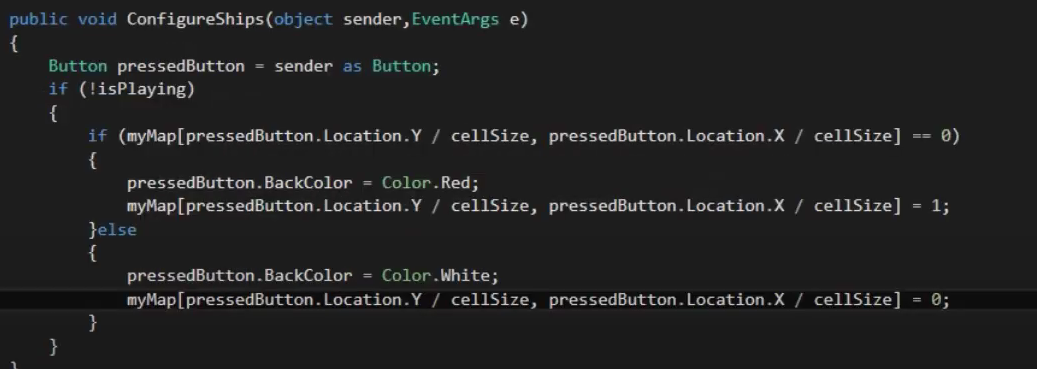
**Рисунок 18. Результат.**



**Рисунок 19. вот так выглядит наш код пока что.**

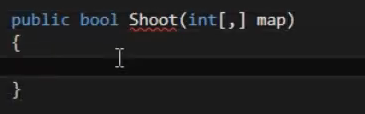


**Рисунок 20. Допишем else в то условие раскраски, чтоб мы не могли красить буквы и цифры и на полях.**

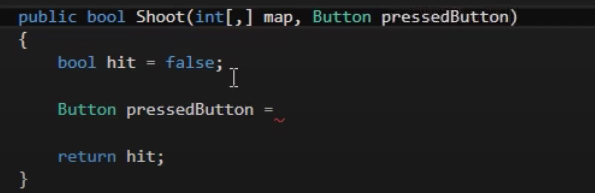


**Рисунок 21. Если ячейка пустая (=0) - заливаем ее красным. Если там уже что-то есть, мы должны ее чистить.**

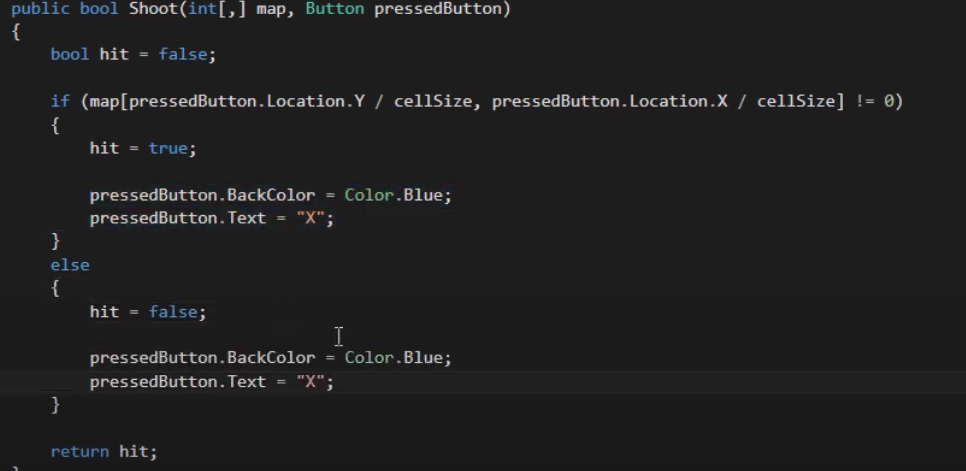
Создаём функцию Shoot, с помощью которой будет осуществлять стрельба по полю противника.



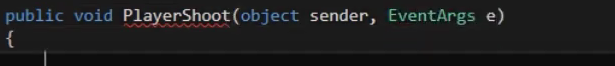
**Рисунок 22. прописываем функцию.**



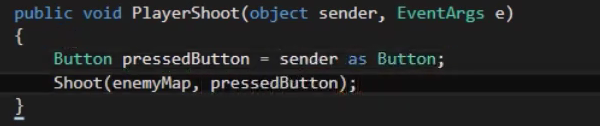
**Рисунок 23. В функцию предадим нажатую кнопку. bool hit = false — если не попали.**



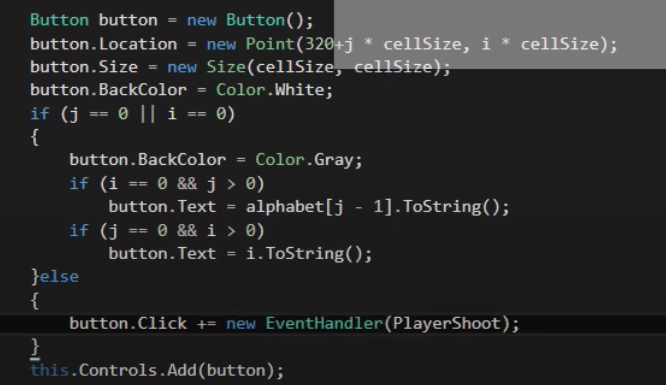
**Рисунок 24. Если мы попали присваиваем синий цвет, если не попали, то чёрный цвет.**



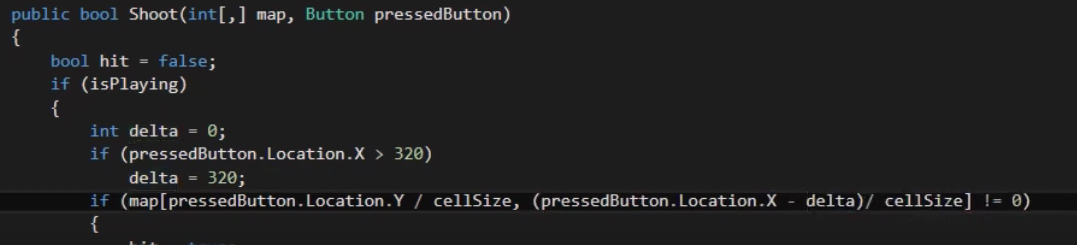
**Рисунок 25. Создаём функцию Player Shoot. Это будет обработчик события нажатия кнопки.**



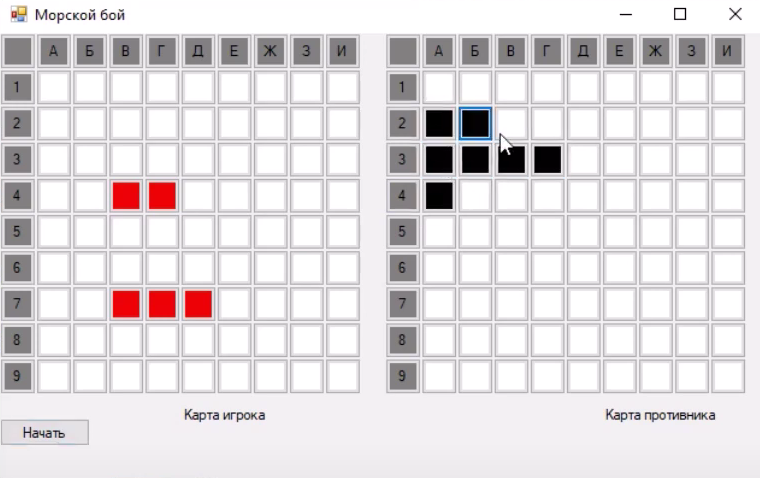
**Рисунок 26. Здесь будем вызывать функцию Shoot (стрельба) и передавать туда карту врага, и нажатую кнопку.**



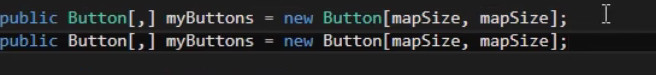
**Рисунок 27. Теперь прописываем кнопке этот обработчик во втором цикле.**



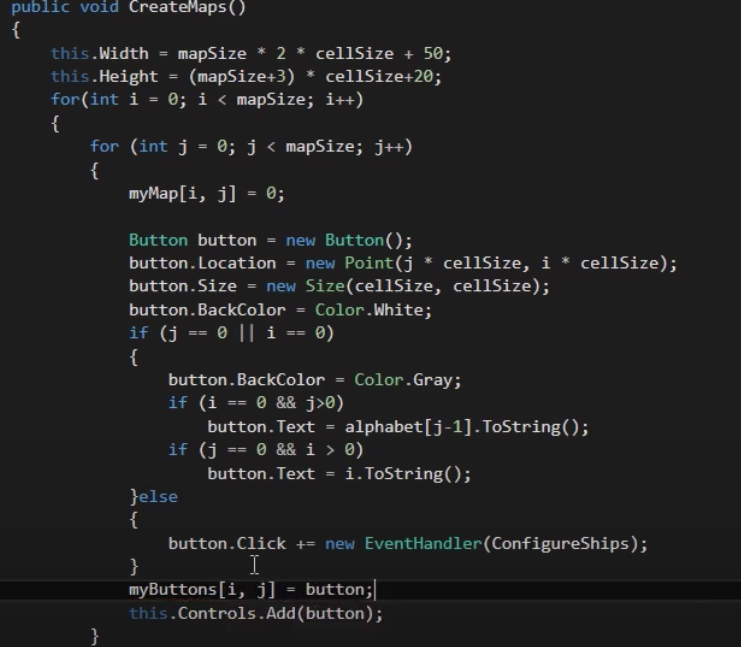
**Рисунок 28. У нас было смещением, и мы должны это учесть в функции Shoot. Карта противника находится со смещением относительно карты игрока.**



**Рисунок 29. Результат.**



**Рисунок 30. Создаём два массива. В них будет храниться действия и ячейки карты игрока и противника.**

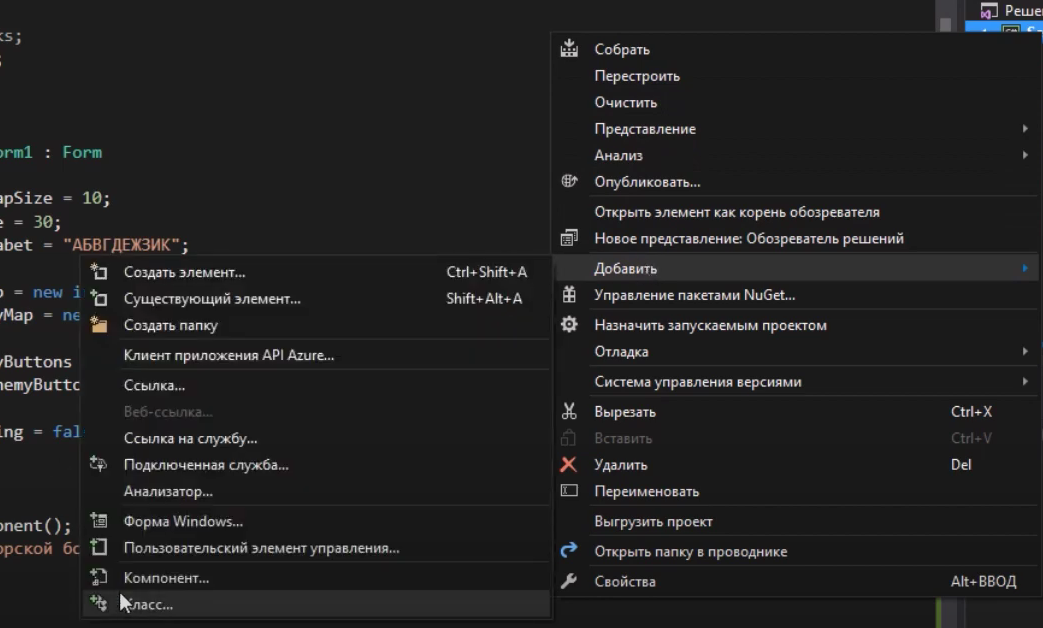


**Рисунок 31. в двух циклах мы вставим добавление созданной кнопки в двумерный массив.**

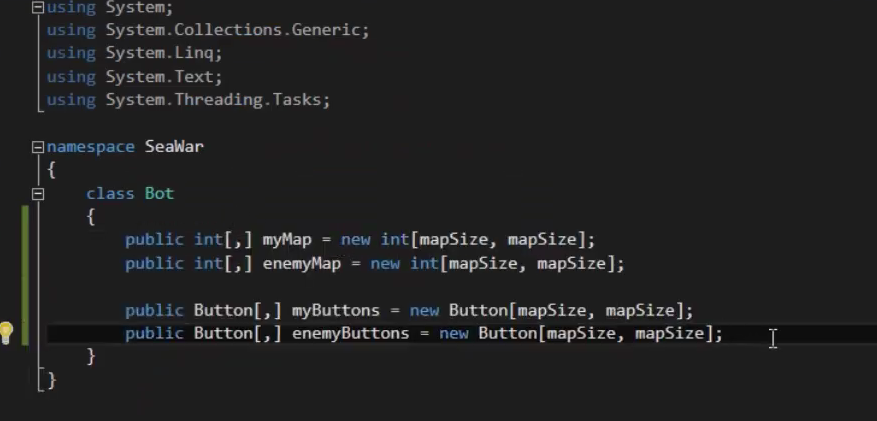


**Рисунок 32.в двух циклах мы вставим добавление созданной кнопки в двумерный массив.**

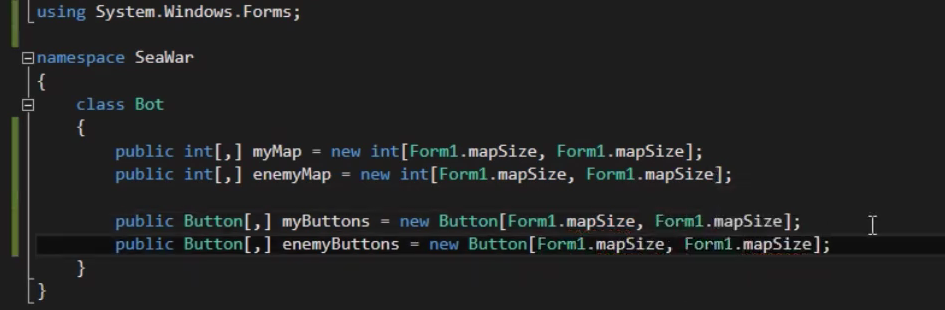
Создаём новый класс, для написания кода для нашего бота.



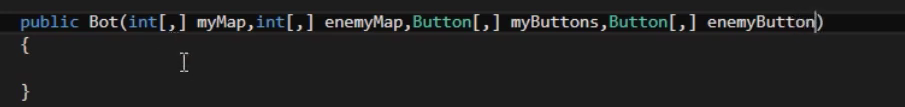
**Рисунок 33. Создание нового класса Bot.cs**



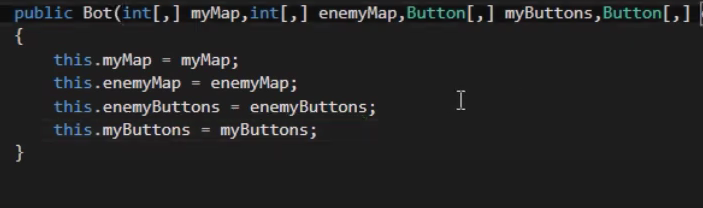
**Рисунок 34. Копируем строчки с основного скрипта. myMap – карта бота, enemyMap -карта игрока.**



**Рисунок 35. Подключаем библиотеку и прописываем формы.**



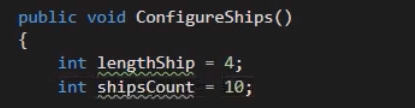
**Рисунок 36. Создание конструктора. Прописываем поле бота и поле игрока.**



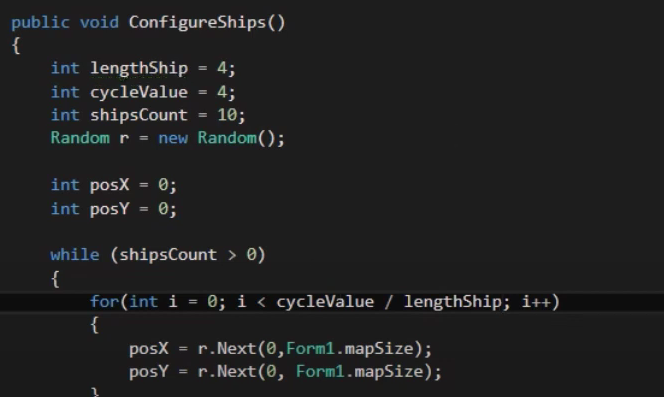
**Рисунок 37. Прописываем определённым переменным их значение.**



**Рисунок 38. Прописываем функцию IsInsideMap, которая будет принимать два параметра. Если какой-либо из параметров выход за границы нашей карты, то мы будем возвращать false. В ином случае true.**



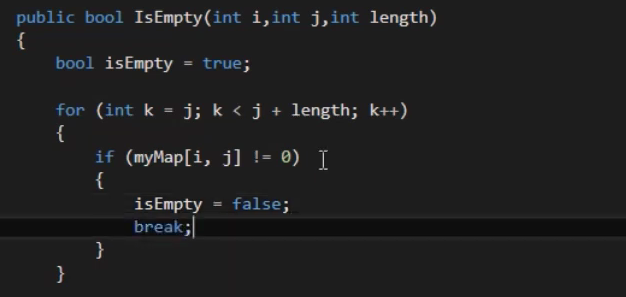
**Рисунок 39. Теперь напишем функцию, которая будет конфигурировать наши корабли для бота. Задаём длину и количество короблей.**



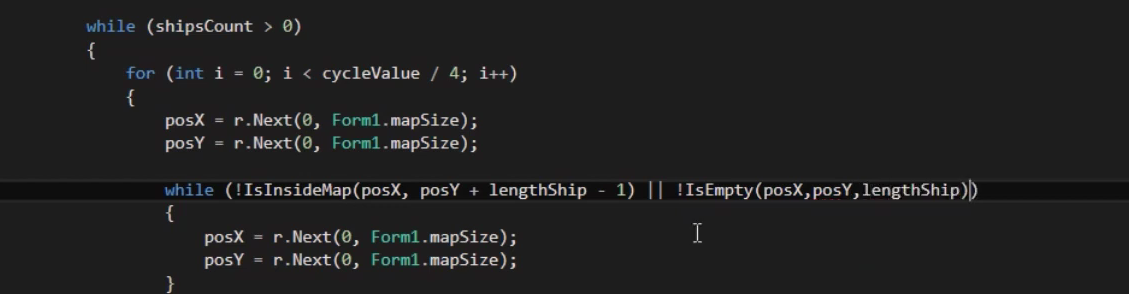
**Рисунок 40. В for будем расставлять корабли определённой длины. posX, posY генерируется рандомно.**



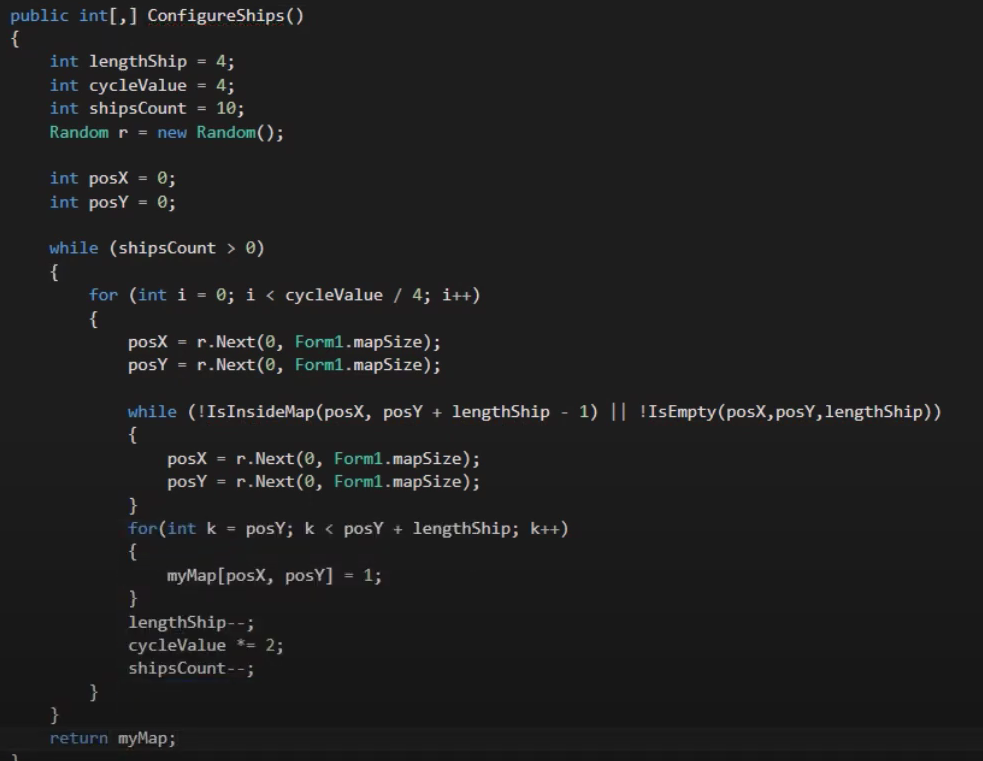
**Рисунок 41. В цикле for через цикл while реализуем расстановку нашего корабля. Если выходит за границы - мы генерируем новый posX. Когда posX и posY уже имеют адекватные значения расставим корабль в этих позициях через еще один цикл for. В конце уменьшаем размер корабля и уменьшаем количество на 1.**



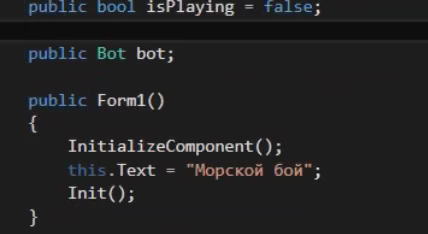
**Рисунок 42. Также пропишем функцию isEmpty, которая будет возвращать нам переменную типа bool. Через нее также будем передавать параметры корабля. В цикле for будем бегать от j, j+ длина до нашего корабля. И соответственно проверять, если у нас элемент не нулевой в карте противника, то isEmpty будет false. Заканчиваем алгоритм.**



**Рисунок 43. Также прописываем это в тот while, где входят ли наши позиции Y и X за границы карты.**



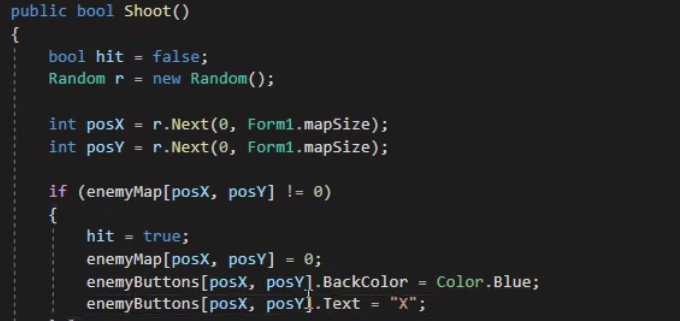
**Рисунок 44. Также изменим нашу функцию, чтоб она возвращала нам двумерный массив.**



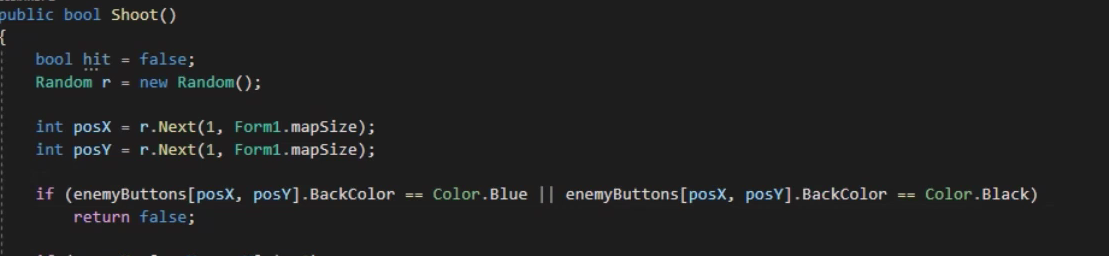
**Рисунок 45. Теперь в основном скрипте пропишем нашего бота.**



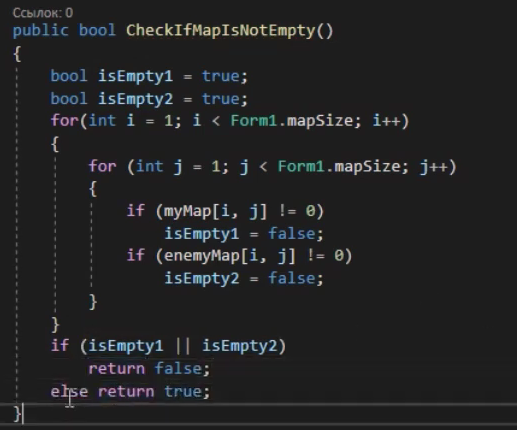
**Рисунок 46. В функцию Init прописываем карту бота, карту игрока, кнопки бота и кнопки игрока. В карту протиника будет записываться возращёенная карта с конфигурацией кораблей.**



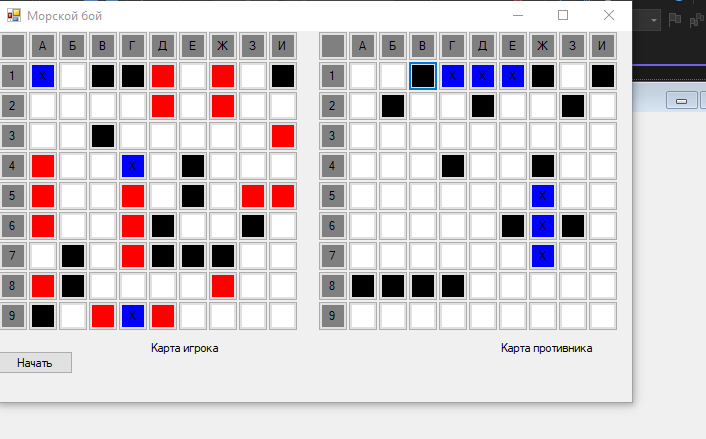
**Рисунок 47. Прописываем функцию Shoot для бота. Нам понадобится генератор рандомнх цифр, также posX и posY в которых как раз будем записывать случайные цифры. Если у нас в этих координатах что-то есть, то hit=true и ячейку бота закрашиваем в синий цвет и Text = “X” – мы попали в корабль.**



**Рисунок 48. Если у нас в текущих координатах цвет повторяется – стрелять не будем.**



**Рисунок 49. Вводи функцию, которая смотрит есть ли у нас ещё живые корабли на картах. Если нет возвращаем false. Если у нас на нашей карте сеть элемента неравный нулю, то в переменную isEmpty = false. И тоже самое пропишем для карты противника. Если карта пустая возвратим false.**



**Рисунок 50. Итоговый результат игрового бота.**

## **Заключение**

Передо мной была поставлена задача создания одной из известных игр Морской бой и отработать навыки написания программного кода игры, и как я считаю задача была выполнена в полном объёме. В дальнейшем для улучшения работоспособности данной игры можно будет сделать ее более простой и понятной для любого пользователя.