Я реализовала класс Ship, который определяет свойства и методы для кораблей. Поля класса: имя корабля, размер корабля, координаты всех клеток корабля (вектор coordinates), направление (вертикальное/горизонтальное). Дополнительно имеются поля количества попаданий (hits) и состояния конкретной клетки корабля (вектор status\_coordinates, где true означает целую клетку, а false — подбитую). В классе Ship реализованы следующие методы:

1. Ship(string name, int size, COORD start, bool is\_vertical). Конструктор. В классе имеется конструктор как с параметрами, так и без (хотя нигде не используетя). Параметрами конструктора являются соответственно имя корабля, его размер, координаты начала корабля, положение (вертикальное/горизонтальное);
2. getCoordinates(). Метод, возвращающий вектор координат всех клеток корабля;
3. isSunk(). Метод, возвращающий true, если корабль потоплен, и false в любом другом;
4. statusShip(). Метод типа int, возвращающий значение, соответствующее состоянию корабля (0 – ранен, -1 – потоплен, 1 - цел);
5. takeHit(int x, int y). Метод обработки попадания. Данный метод в качестве входных параметров принимает координаты выстрела по X и Y. Если по данным координатам находится клетка корабля, метод меняет состояние этой клетки на false (подбита), и возвращает true (есть попадание). В ином случае метод вернёт false (промах).
6. ~Ship(). Деструктор.

Для облегчения работы других классов заголовочный файл класса Ship содержит структуру ShipType. Она содержит в себе поля: имя типа корабля, размер, соответствующий типу, и количество кораблей этого типа.

Мною также был реализован класс Board. Класс имеет поля: размерность поля, игровое поле, представленное в виде матрицы символов (вектора векторов символов), вектор Ship размещённых кораблей. Класс Board содержит следующие методы:

1. Board(int size). Конструктор. В качестве входных параметров использует соответственно размерность игрового поля;
2. placeShip(const Ship& ship). Метод размещающий корабль на поле grid, заменяя символы пробела (пустые клетки) на символы S (ship), по координатам корабля (объекта класса Ship), который передаётся в метод по ссылке;
3. placeDeadField(const Ship& ship). Метод, размещающий «мёртвую зону» вокруг (предполагается, что потопленного) корабля (объект класса Ship) на поле grid, заменяя символы пробела (пустые клетки) на символы О (мертвые клетки). Корабль передаётся в метод по ссылке;
4. display(bool second\_battlefield, const Board& second\_board). Отображение двух полей (своего и чужого), используется в случае игры против ИИ. В качестве входных параметров используются булевая переменная, говорящая о наличии второго поля, и само поле противника (объект класса Board), передающееся по ссылке. В зависимости от хранимого в grid символа клетки в консоли будут отображаться разными цветами, целые клетки кораблей на поле противника скрываются;
5. display(const Board& second\_board, string titul\_player). Отображение поля противника (со скрытыми кораблями), используется в случае игры двух людей. Метод получает по ссылке поле противника (объект класса Board) и идентификатор атакующего игрока.
6. display\_ship\_on\_board(const Ship& ship, int x, int y). Метод, написанный для удобства работы с другими классами. Отображает корабль на доске перед размещением (как корабль будет стоять на доске в случае размещения), путём окрашивания в красный цвет клеток. В качестве входных параметров метод получает корабль (объект класса Ship) по ссылке и координаты начала корабля по X и Y;
7. can\_place\_ship(const Ship& ship). Метод проверяющий выполнение всех условий для размещения корабля на поле (корабль не должен выходить за пределы поля, все соседние клетки должны быть свободны). Корабль (объект класса Ship) передаётся в метод по ссылке. Если хотя бы одно условие не выполняется метод возвращает false, в ином случае метод возвращает true;
8. void setColorWithBackground(int textColor, int backgroundColor). Метод, устанавливающий атрибуты выводимого на консоль текста. Метод принимает значение цвета текста и фона соответственно.
9. clearPartOfConsole(int startLine, int endLine). Метод, стирающий все записи в консоли в заданном диапазоне строк. В качестве входных параметров принимает номер начальной и конечной строки.
10. SetCursor(int x, int y). Метод перемещающий курсор на заданные координаты.
11. ~Board(). Деструктор.
12. processShot(int x, int y). Метод обработки выстрела. Данный метод в качестве входных параметров принимает координаты выстрела по X и Y соответственно. Для работы этого метода и для удобства работы с другими классами была создана структура PlayerResultOfShot содержащая поля: rezult\_of\_shot – результат выстрела (1 – попадание, 0 – промах, -1 – обстрел мертвой или уже подбитой клетки), size\_of\_ship – размер корабля, ship\_dead – потоплен ли корабль (true – потоплен, false – не потоплен), damage\_more\_one\_square – повреждено ли больше одной палубы корабля (true – повреждений больше 1, false – повреждений 0 или 1). Метод возвращает данную структуру. В самом же методе осуществляется проверка, осуществлено ли попадание по кораблю, далее происходит поиск подбитого корабля, заполнение полей структуры и размещение «мёртвой зоны» в случае затопления корабля;