**КОМПЬЮТЕРНАЯ АКАДЕМИЯ «ШАГ»**

**Днепропетровский филиал**

**Кафедра Разработки программного обеспечения**

**КУРСОВОЙ ПРОЕКТ**

**Тема:** Программирование консольного приложения «Морской бой»

**Студента группы: ЕКО 15 – П - 1  
Ус В.С.**

**Научный руководитель:  
Шаптала М.В.**

Днепропетровск 2016

**Содержание**

**Введение** Аннотация   
 Правила игры  
 **1. Проектирование классов**  
 1.1 Ship (Корабль)   
 1.2 Player (Игрок)   
 1.3 Game (Игра)   
 1.4 PREDICAT\_downed\_ship (предикат подбитого корабля)   
 **2. Описание классов, функций и др. вне классов**  
 2.1 Описание классов  
 2.2 Описание функций вне классов  
 2.3 Defines и enum

**3. Интерфейс программы**  
**4. Диаграммы (UML и Graph)**

4.1 UML 4.2 Graph **Выводы  
5. Список литературы**

**Введение**

*Аннотация*

Данная программа была написана на языке **С++** в среде Visual Studio, с целью изучения, закрепления, углубления и расширения теоретических знаний, практических умений и навыков в соответствии с учебными планами. В ходе работы было написано консольное приложение «Морской бой».

*Правила игры*

**Морской бой -** игра происходит на поле 10х10 клеточек каждого игрока, на котором размещается флот кораблей. Флот состоит из:

\* 1 корабль - ряд из 4 клеток четырёхпалубные

\* 2 корабля - ряд из 3 клеток трёхпалубные

\* 3 корабля - ряд из 2 клеток двухпалубные

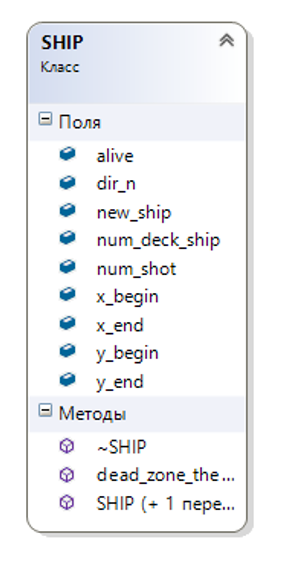
\* 4 корабля - ряд из 1 клеточки однопалубные

При размещении корабли не могут касаться друг друга углами. Палубы кораблей надо строить «в линейку», а не изгибами. Главное: нельзя строить палубы одного корабля по диагонали. Перед началом боевых действий игроки бросают жребий или договариваются, кто будет ходить первым. Игрок, выполняющий ход, совершает выстрел — называет вслух координаты клетки, в которой, по его мнению, находится корабль противника, например, «A1». Если выстрел пришёлся в клетку, не занятую ни одним кораблём противника, то следует ответ «Мимо!» и стрелявший игрок ставит на чужом квадрате в этом месте точку. Право хода переходит к сопернику. Если выстрел пришёлся в клетку, где находится много трубный корабль (размером больше чем 1 клетка), то следует ответ «Ранил!». Стрелявший игрок ставит на чужом поле в эту клетку крестик, а его противник ставит крестик на своём поле также в эту клетку. Стрелявший игрок получает право на ещё один выстрел. Если выстрел пришёлся в клетку, где находится однопалубный корабль или последнюю непоражённую клетку многопалубного корабля, то следует ответ «Потоплен!» или «Убит!». Оба игрока отмечают потопленный корабль на листе. Стрелявший игрок получает право на ещё один выстрел. Победителем считается тот, кто первым потопит все 10 кораблей противника.

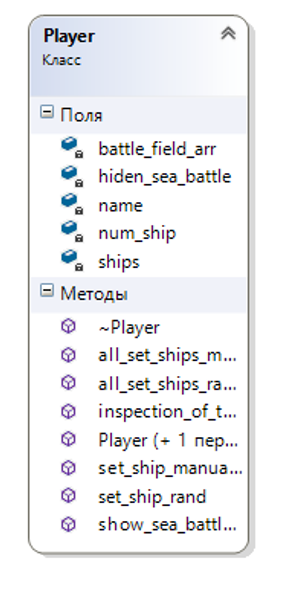
1. **Проектирование классов**

В данной программе использовалось 4 собственноручно написанных класса:

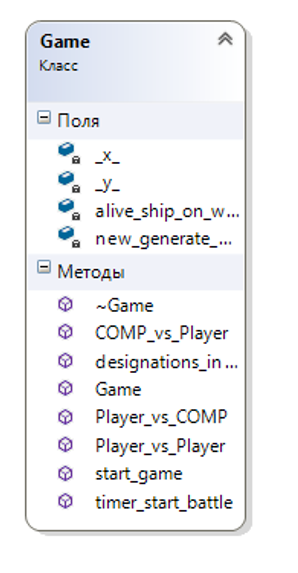
*Класс SHIP* - отвечает за создание и описание кораблей игрока.



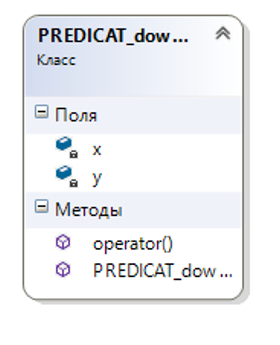
*Класс Player -* отвечает за создание кораблей игрока и управления ими



*Класс GAME* - отвечает за управление игрой

**

*Класс PREDICAT\_downed\_ship* – возвращает координаты попадания по кораблю



1. **Описание классов и функций**

*Описание классов*

*Класс SHIP*

*Поля:*

1. int x\_begin - координата начала корабля по строке
2. int x\_end – координата конца корабля по строке
3. int y\_begin – координата начала корабля по столбцу
4. int y\_end – координата конца корабля по столбцу
5. bool alive – состояние корабля (изначально true)
6. int num\_deck\_ship – количество палуб корабля
7. char \*new\_ship – позволяет динамически создать корабль с нужным количеством палуб
8. direction dir\_n – определяет положение корабля на поле (VERTICAL/HORIZONTAL)
9. int num\_shot – считает количество попаданий по кораблю

*Методы:*

1. void dead\_zone\_the\_ship (char arr [][COLUMNS], bool \_is\_dead) - отвечает за обрисовку «Мертвой зоны» корабля, т.е. при убийстве корабля вокруг на ячейку ++ появляется бомба, что означает: данное место уже поражено попаданием
2. friend ostream& operator << (ostream&stream, const SHIP& obj) – перегрузка оператора вывода для необходимости вывода информации о кораблях игрока

*Класс Player*

*Поля:*

1. char battle\_field\_arr[10][10] – поле боя по которому производятся выстрелы противника
2. char hiden\_sea\_battle[10][10] – скрытое поле боя на котором расстановлены корабли игрока
3. string name – имя игрока
4. int num\_ship – общее количество кораблей игрока
5. vector<SHIP> ships – вектор держит в себе все корабли игрока и информацию о них

*Методы:*

1. void show\_sea\_battle\_hiden\_player() – позволяет вывести скрытое поле боя игрока на котором находятся корабли игрока
2. vector<SHIP> set\_ship\_rand (int size\_deck) – позволяет рендомно установить на поле один корабль с указанным количеством палуб. При выполнении метода созданный корабль помещается в вектор.
3. bool inspection\_of\_the\_ship\_at\_the\_touch(int x, int y, direction dir, int size\_deck) – при установке корабля методом set\_ship\_rand позволяет проверить смежные клетки корабля.
4. void all\_set\_ships\_rand() – позволяет рендомно установить все корабли на скрытом поле боя
5. vector<SHIP> set\_ship\_manually (int size\_deck) – позволяет вручную установить на поле один корабль с указанным количеством палуб. При выполнении метода созданный корабль помещается в вектор.
6. void all\_set\_ships\_manually() – позволяет вручную установить все корабли на скрытом поле боя
7. void smart\_game\_comp()
8. friend ostream& operator << (ostream&stream, Player& obj) – перегрузка вывода позволяет вывести скрытое поле боя игрока.
9. friend istream& operator >> (istream&stream, Player& obj) – перегрузка ввода позволяет ввести имя игрока

*Класс GAME:*

# Поля: bool new\_generate\_koordinat; //для позволения генерации новых координат int \_x\_; //строка 1 >< 10 int \_y\_; //столбец A >< J bool alive\_ship\_on\_which\_the\_last\_hit; //для определения жизни корабля по которому был последний выстрел, чтобы определить продолжать ли стрелять компу по этому кораблю

*Методы:*

1. void Player\_vs\_Player (Player player1OBJ, Player player2OBJ) – описывается игра Игрок против Игрока
2. void COMP\_vs\_Player (Player compOBJ, Player playerOBJ) – описывается игра Компьютер против Игрока
3. void Player\_vs\_COMP (Player playerOBJ, Player compOBJ) – описывается игра Игрок против Компьютера
4. void start\_game() – описывается интерфейс меню

*Описание функций вне классов*

1. void MyColor (int text, int background) – устанавливает цвет текста и фона

*Отрисовка поля:*

1. void put\_top\_line (ostream& stream) – выводит верхнюю строку
2. void put\_middle\_line (ostream& stream) – выводит ячейки поля и строки между ними
3. void put\_bottom\_line (ostream& stream) – выводит на экран нижнюю строку
4. ostream& battlefield (ostream& stream, char arr[][COLUMNS], char \*name) – выводит полное поле на экран. В данном методе происходит раскраска поля

*Defines и enum*

*Defines*

1. #define ROWS 10 – количество строк поля
2. #define COLUMNS 10 – количество столбцов поля
3. #define ship char(251) – обозначение корабля
4. #define bomb char253 – обозначение бомбы
5. #define hit char(229) – обозначение поражения корабля
6. #define tag\_sea char(177) – обозначение пустых ячеек поля

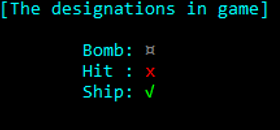
Enum

enum ConsoleColor – держит в себе 16 цветов и позволяет при установке цвета текста и фона, в функцию MyColor передать аргументы в качестве названия цвета

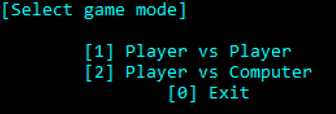
1. Black – черный
2. Blue – синий
3. Green – зеленый
4. Cyan – темно – бирюзовый
5. Red – красный
6. Magenta – пурпурный
7. Brown – коричневый
8. Light Gray – светло - серый
9. Dark Gray – темно – серый
10. Light Blue – светло – синий
11. Light Green – светло - зеленый
12. Light Cyan – светло – бирюзовый
13. Light Red – светло – красный
14. Light Magenta – светло – пурпурный
15. Yellow – желтый
16. White – белый

**4. Интерфейс программы**

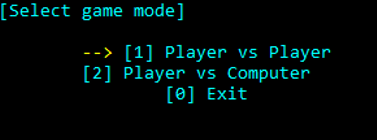
*Обозначения в игре:*

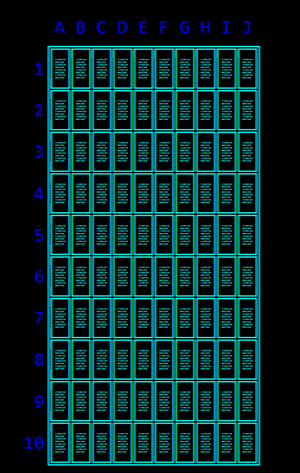
**

*Меню:*

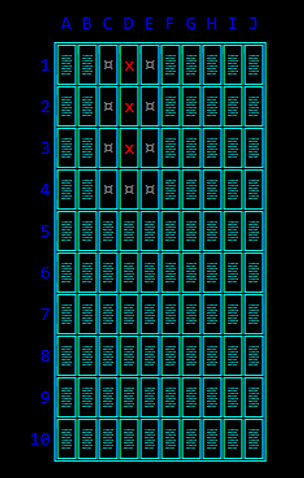
**

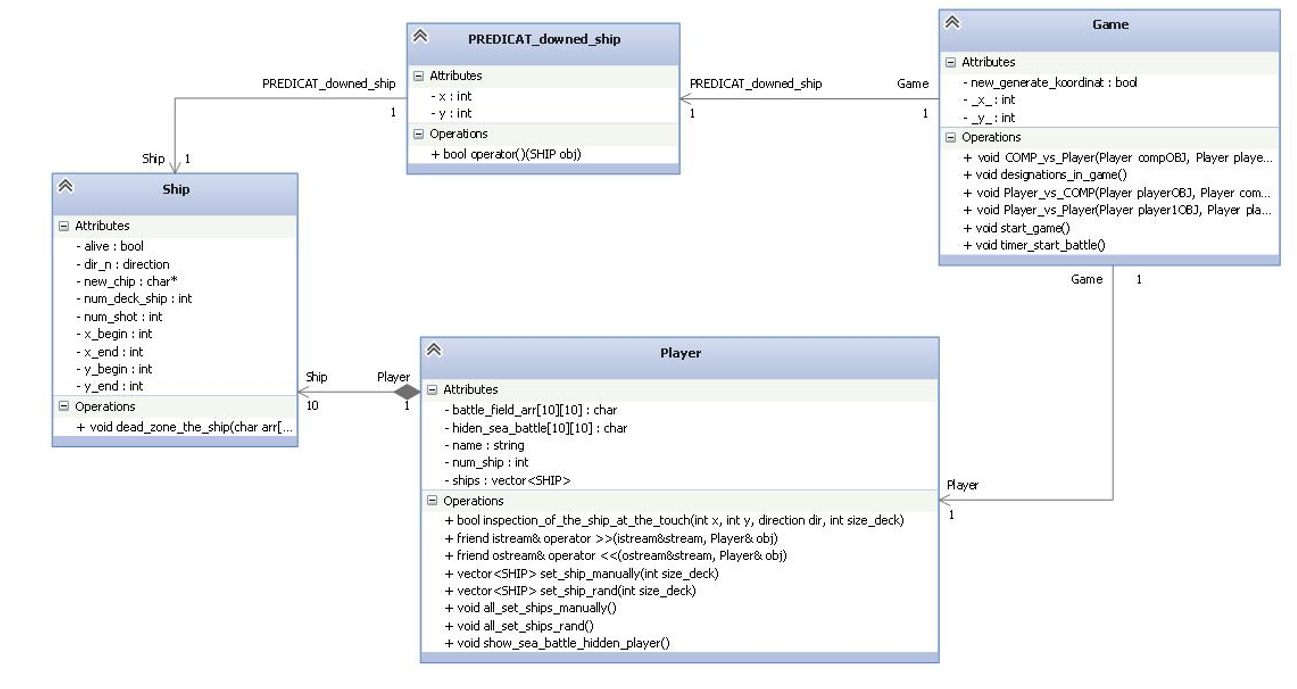
*Выбор в меню:*

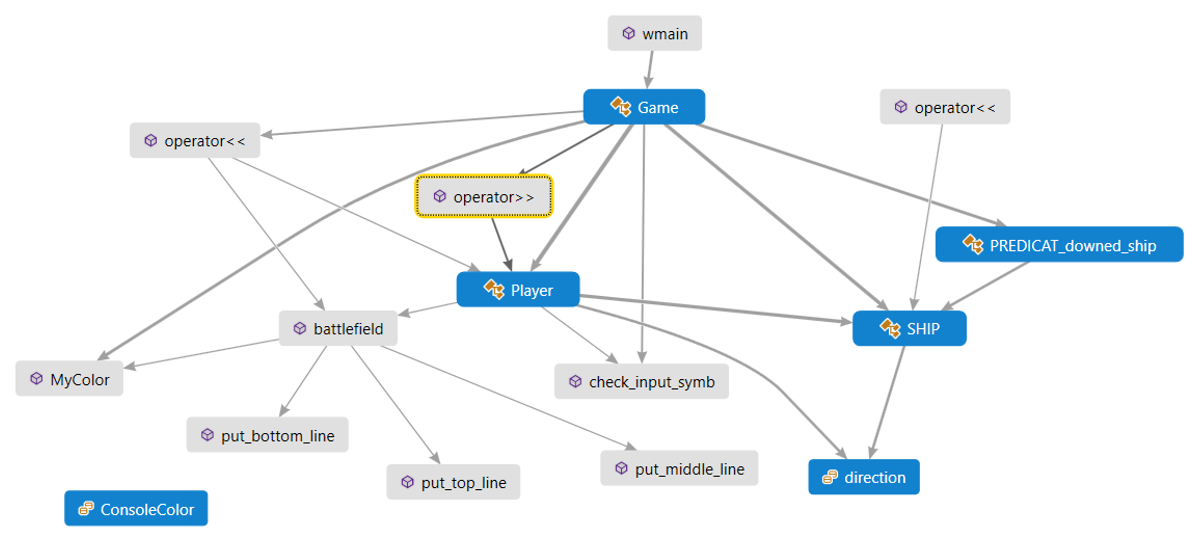
**

*Поле боя:  
*

*Убийство корабля:*

******





**Выводы**

По окончанию написания консольного приложения «Морской бой», мною были закреплены и расширены знания языка программирования С++ в среде разработки Visual Studio. За период написания курсового проекта я овладел навыками самостоятельной разработки консольного приложения.

**Список литературы**

1. Написание функции MyColor (int text, int background). Данная функция предназначена для изменения цвета интерфейса программы. Была написана с помощью:

<http://www.cyberforum.ru/cpp-beginners/thread52526.html>

1. Отрисовка поля была выполнена при помощи ASCII таблицы символов:

<http://www.asciitable.com/>