Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное   
учреждение высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

**Отчет по учебной практике второго семестра**

**Выполнил**:студент группы 381703-2

Родионов Денис Михайлович

**Проверил**:

Доцент кафедры МОСТ, к.т.н.

Сысоев А.B.

Нижний Новгород

2018

Содержание

[Введение 3](#_Toc270962758)

[Лабораторная работа №1. Конвертер весов 4](#_Toc270962759)

[Лабораторная работа №2. Строка 8](#_Toc270962759)

[Лабораторная работа №3. Формула Тейлора 15](#_Toc270962759)

[Лабораторная работа №4. Песенник 21](#_Toc270962759)

[Лабораторная работа №5. Билетная касса 27](#_Toc270962759)

[Лабораторная работа №6. Игра «Морской бой» 33](#_Toc270962759)

[Заключение 41](#_Toc270962765)

[Литература 42](#_Toc270962766)

[Приложения 43](#_Toc270962767)

# Введение

Во втором семестре мы познакомились с ключевым понятием – Объектно-ориентированное программирование. Почему объектно-ориентированному программированию отдается предпочтение в большинстве проектов? ООП предлагает эффективный способ борьбы со сложностью. Вместо того чтобы рассматривать программу как последовательность исполняемых инструкций, оно представляет ее как группу объектов с определенными свойствами и производит с ними определенные действия. На практике предлагалось рассмотреть несколько реализаций программ, используя ООП, без которого модель и план решения задачи оказались бы сложнее.

**Лабораторная работа №1. Конвертер весов.**

# Постановка задачи

Требуется разработать класс конвертер весов. Класс должен хранить вес в килограммах и предоставлять методы по его преобразованию в другие единицы измерения (аптечный фунт, тройская унция, пуд).

Класс должен предоставлять операции:

1) установить текущий вес в килограммах,

2) узнать текущий вес в килограммах,

3) узнать текущий вес в выбранной единице измерения (из списка поддерживаемых).

Помимо этого, класс должен содержать все необходимые конструкторы, оператор присваивания, а также «уметь» выводить себя на консоль.

# Руководство пользователя

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++.

На первом этапе работы программы пользователю необходимо ввести вес в килограммах, создав тем самым объект класса конвертер весов.

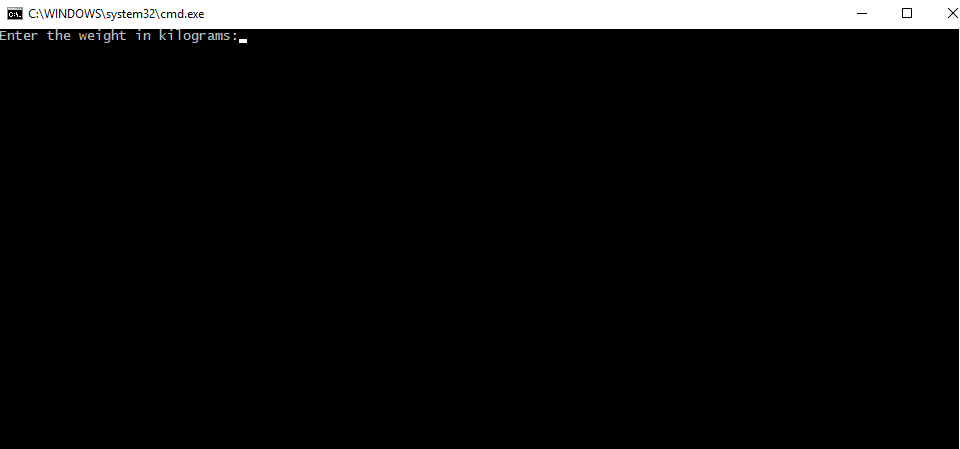


Рис.1. Старт программы

После этого пользователю необходимо выбрать одно из трех действий, предлагаемых программой:

1. Вывести введенный вес в килограммах в консоль.
2. Вывести вес, переведенный из килограмм в другую единицу измерения, реализованную в программе, а именно: аптечный фунт, тройская унция, пуд.
3. Изменить вес, введенный в начале в килограммах

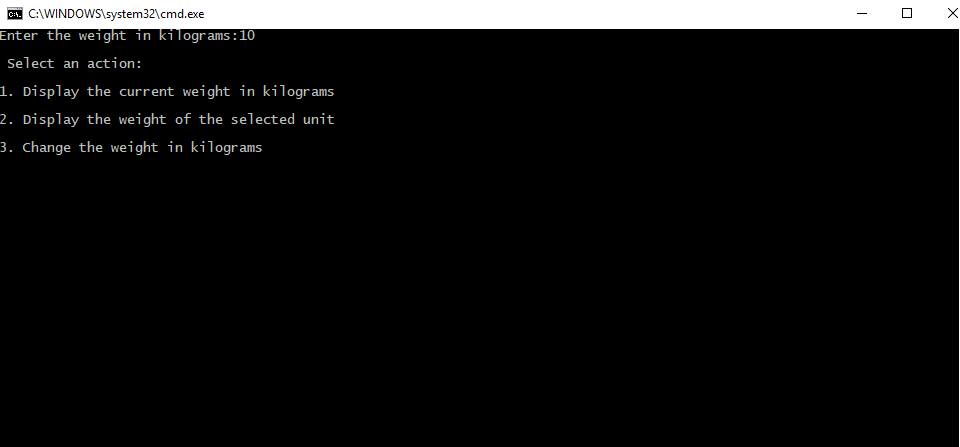


Рис.2. Предлагаемые действия программы

Выбрав, к примеру, вывод на экран веса в другой единице измерения, программа предложит три варианта того, в каком счислении пользователь желает вывести вес.

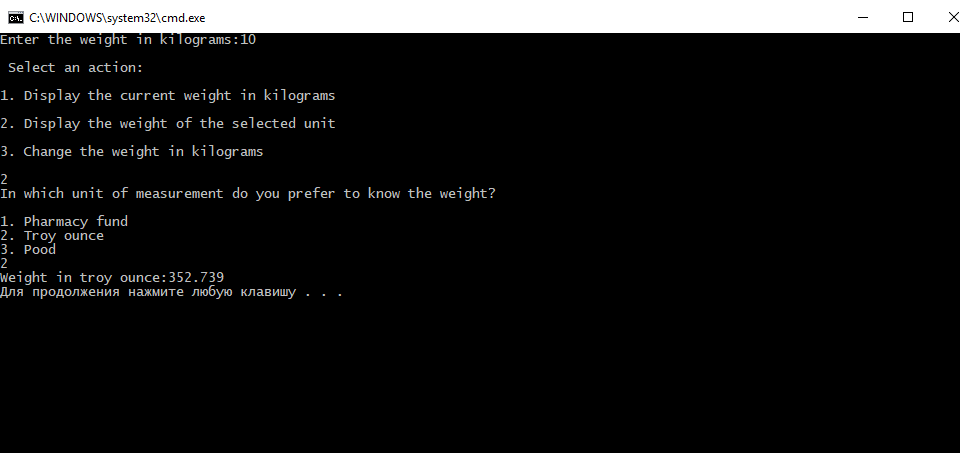


Рис.3.Вывод веса в тройской унции.

# Руководство программиста

В программе реализован один класс под названием «Конвертер весов». В этом классе одно приватное поле – переменная типа int, отвечающая за вес в килограммах. Помимо этого, в классе имеются следующие методы:

1. Все необходимые служебные методы,
2. Метод, изменяющий введенный вес в килограммах,
3. Методы, печатающие текущий вес в килограммах, измененный вес в аптечном фунте, в тройской унции, в пуде,
4. Методы, возвращающие текущий вес в килограммах, измененный вес в аптечном фунте, в тройской унции, в пуде.

## **Описание структуры программы**

Программа построена на использовании интуитивно понятного пользователю меню.

В основная части главной функции программы main создается объект типа «Конвертер весов», проверяется корректность оператора присваивания, реализованного в классе. Главная функция представляет собой оператор множественного выбора swich. В зависимости от выбора пользователем действия, вызывается тот или иной метод класса «Конвертер весов».

## 

## **Описание алгоритмов**

1. Метод, изменяющий текущий вес в килограммах.

Данный метод реализован как функция типа void. Она принимающая параметр типа double и присваивает его полю, отвечающему за вес в килограммах.

1. Методы, печатающие текущий вес в килограммах, измененный вес в аптечном фунте, в тройской унции, в пуде.

Данные методы схожи и реализованы как функции типа void. Каждая из них не принимает никаких параметров и с помощью потока вывода cout выводят соответствующую единицу измерения.

1. Методы, возвращающие текущий вес в килограммах, измененный вес в аптечном фунте, в тройской унции, в пуде.

Данные методы схожи и реализованы как функции, возвращающие значение типа double.

Каждая из них не принимает никаких параметров и с помощью ключевого слова return возвращает соответствующую единицу измерения.

**Лабораторная работа №2. Строка.**

# Постановка задачи

Требуется разработать класс «Строка», который должен хранить строку символов произвольной длины.

Класс должен предоставлять следующие операции:

1) задать строку,

2) узнать длину строки,

3) получить символ строки по его индексу,

4) изменить символ строки по заданному индексу,

5) выделить подстроку из строки,

6) проверить, является ли строка палиндромом,

7) найти, сколько разных символов латинского алфавита содержится в строке.

Класс должен содержать все необходимые конструкторы, деструктор, оператор присваивания, а также «уметь» выводить себя на консоль.

# Руководство пользователя

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++.

На первом этапе работы программы пользователю необходимо ввести саму строку, с которой дальше предстоит работать.

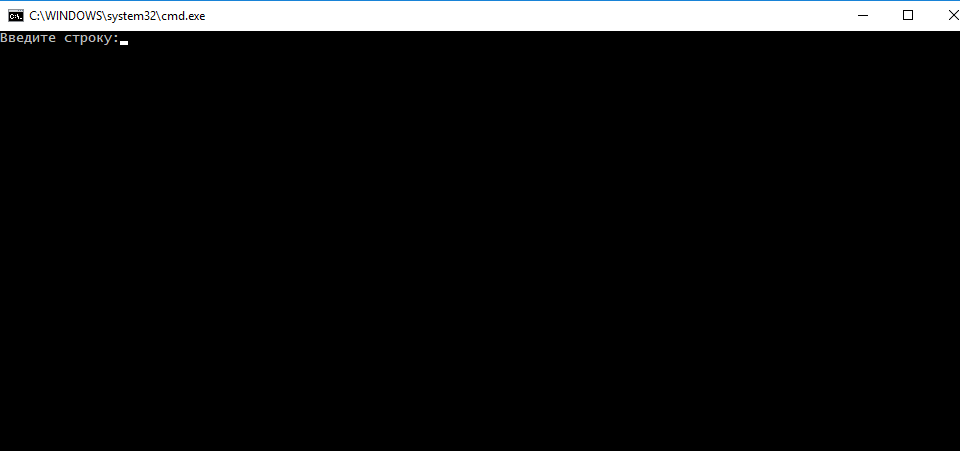


Рис.1. Старт программы.

После этого пользователю предлагается выбрать одно из семи действий, предлагаемых программой:

1. Вывести строку на экран,
2. Проверить строку на палиндром,
3. Узнать количество разных символов в строке,
4. Выделить подстроку из строки,
5. Узнать длину строки,
6. Получить символ строки по его символу,
7. Изменить строку по заданному символу.

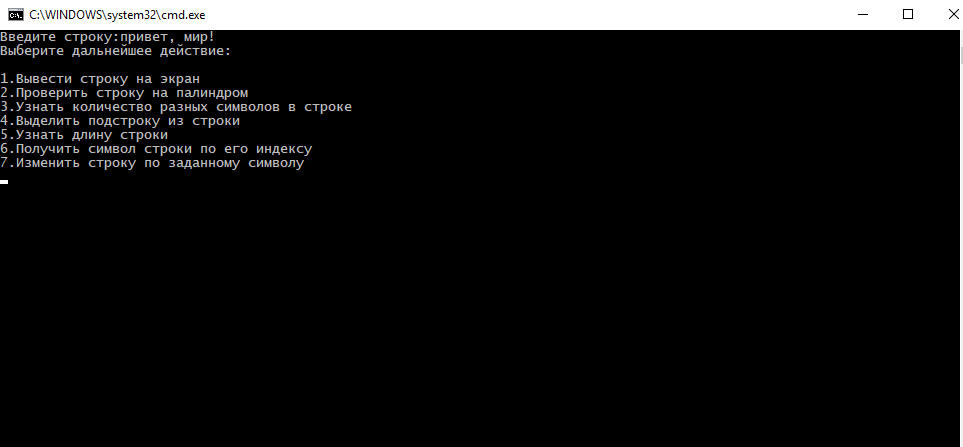


Рис.2. Предлагаемые действия программы.

Разберёмся со всем по порядку. Нажав цифру «1», пользователь увидит ту строку, которую ввёл.



Рис.3. Действие первого пункта программы.

Нажав «2», пользователь получит ответ на вопрос, является ли строка палиндромом.

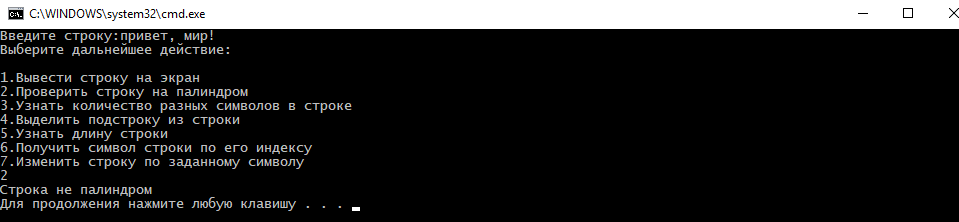


Рис.4. Действие второго пункта программы.

Чтобы узнать количество разных символов в строке, пользователю предстоит нажать «3».

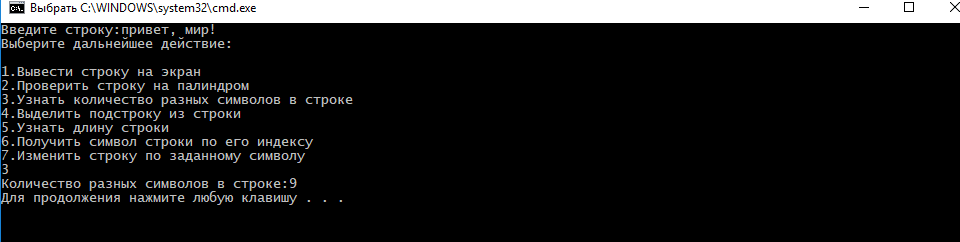


Рис.5. Действие третьего пункта программы.

Чтобы выделить подстроку из строки, следует нажать «4».

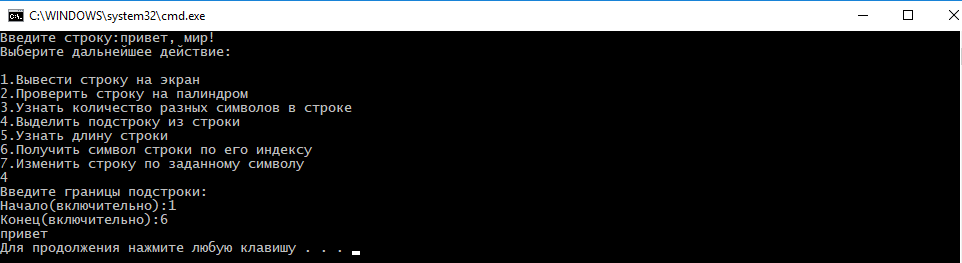


Рис.6. Действие четвертого пункта программы.

Нажав «5», пользователь узнает, какова длина той строки, которую он ввел.

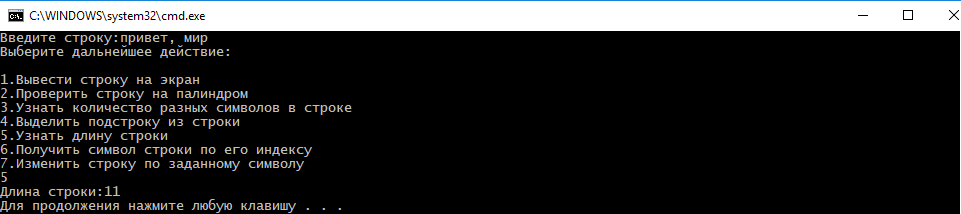


Рис.7. Действие пятого пункта программы.

Нажав «6», пользователь получит символ строки по его индексу.



Рис.8. Действие шестого пункта программы.

И, наконец, нажав «7», пользователь может явно изменить символ по заданному номеру.

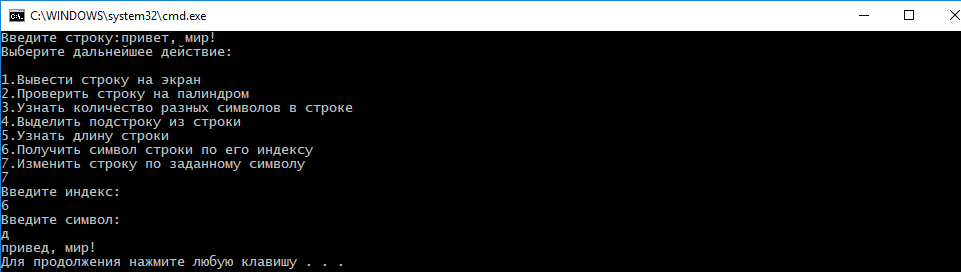


Рис.9. Действие седьмого пункта программы.

# Руководство программиста

В программе реализован один класс под названием «Строка». В этом классе два приватных поля – переменная типа int, отвечающая за длину строки и указатель на тип данных char, отвечающий за место в памяти, где хранится сама строка. Помимо этого, в классе имеются следующие методы:

1. Все необходимые служебные методы: конструктор, деструктор, конструктор копирования оператор присваивания,
2. Метод, отвечающий за вывод текущей строки на экран,
3. Метод, задающий строку,
4. Метод, возвращающий длину строки,
5. Метод, который получает символ строки по его индексу,
6. Метод, изменяющий строку по заданному символу,
7. Метод, проверяющий строку на палиндром,
8. Метод, узнающий сколько различных символов, содержится в строке.

## **Описание структуры программы**

Программа построена на использовании интуитивно понятного пользователю меню.

В основной части главной функции программы main создается два объекта типа «Строка», проверяется корректность оператора присваивания, реализованного в классе. Главная функция представляет собой оператор множественного выбора swich. В зависимости от выбора пользователем действия, вызывается тот или иной метод класса «Строка».

## **Описание алгоритмов**

1. Метод, отвечающий за вывод текущей строки на экран.

Данный метод реализован как функция типа void. Она не принимает никаких параметров и состоит из цикла for, который посимвольно выводит текущую строку на экран.

1. Метод, задающий строку.

Метод реализован как функция типа void. Он принимает один параметр, а именно: указатель на тип данных char. В зависимости от длины строки выделяется память под указатель, который находится в поле класса и запоминается её длина в переменную, также хранящуюся в поле. После этих действий строка копируется.

1. Метод, возвращающий длину строки

Метод реализован как функция, которая возвращает тип данных int. Он не принимает никаких параметров и состоит их одного ключевого слова return.

1. Метод, который получает символ строки по его индексу.

Метод возвращает тип данных char и принимает переменную типа int. С помощью операции разыменования у строки возвращается именно то, что нужно.

1. Метод, изменяющий строку по заданному символу

Метод реализован как функция типа void, принимающая номер символа и сам символ. Аналогично с помощью операции разыменования указателя строке на место переданного номера присваивается переданный символ.

1. Метод, проверяющий строку на палиндром

Метод возвращает тип bool и не принимает никаких параметров. Последовательно выполняются следующие процедуры: создаются два указателя на тип char, одному присваивается начало строки, другому – её конец. С помощью цикла for указатели сравнивают соответствующие значения. Если строка оказалась палиндромом, вернется true, иначе – false.

1. Метод, узнающий сколько различных символов, содержится в строке.

Метод возвращает переменную типа int и не принимает параметров. В блоке метода создается переменная, которая со временем изменяется. С помощью цикла for отдельные символы строки сравниваются со всеми предыдущими. Если совпадающих нет, созданная переменная увеличивается, иначе – сохраняет свое предыдущее значение. После выхода из цикла именно она возвращается методом.

**Лабораторная работа №3. Формула Тейлора.**

# Постановка задачи

Требуется разработать класс «Ряд Тейлора», который должен формировать ряд Тейлора для выбранной функции из списка поддерживаемых с заданным числом членов ряда. Список функций: sin(x), cos(x), exp(x).

Класс должен содержать необходимые служебные методы (конструкторы, деструктор и пр.).

Класс должен предоставлять следующие операции:

1) задать текущую функцию,

2) узнать текущую функцию,

3) задать текущее число членов ряда,

4) узнать текущее число членов ряда,

5) выдать формулу ряда для выбранной функции,

6) выдать значение заданного члена ряда,

7) рассчитать значение ряда в выбранной точке x,

8) вывести отклонение значения ряда в выбранной точке от эталонного значения текущей функции в данной точке (эталонное значение вычисляется, используя соответствующую функцию из стандартной библиотеки C++).

# Руководство пользователя

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++.

На первом этапе работы программы пользователю необходимо выбрать функцию, в которой он желает работать в дальнейшем.



Рис.1. Старт программы.

После этого пользователю предлагается поочередно ввести точку вычисления и количество членов ряда. После этого ему необходимо выбрать одно из семи действий:

1. Узнать функцию,
2. Узнать число рядов,
3. Узнать точку вычисления,
4. Вывести формулу ряда,
5. Узнать значение конкретного члена ряда,
6. Узнать значение ряда в выбранной точке,
7. Рассчитать отклонение от эталонного значения.

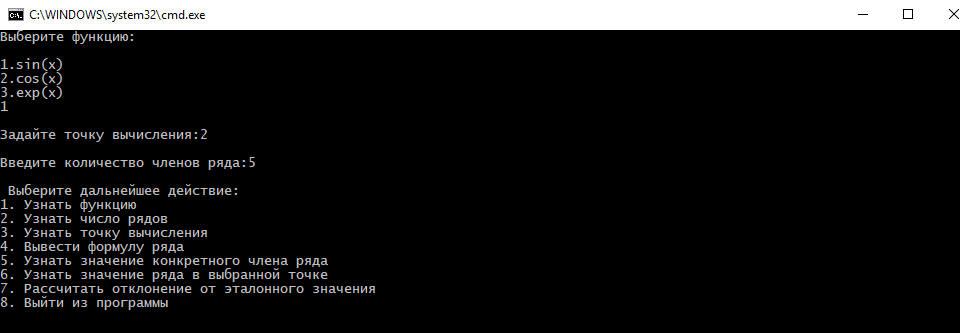


Рис.2. Предлагаемые действия программы.

Разберёмся со всем по порядку. Нажав цифру «1», «2» или «3» пользователь соответственно узнает функцию, с который работает, число рядов или точку вычисления. Пример:

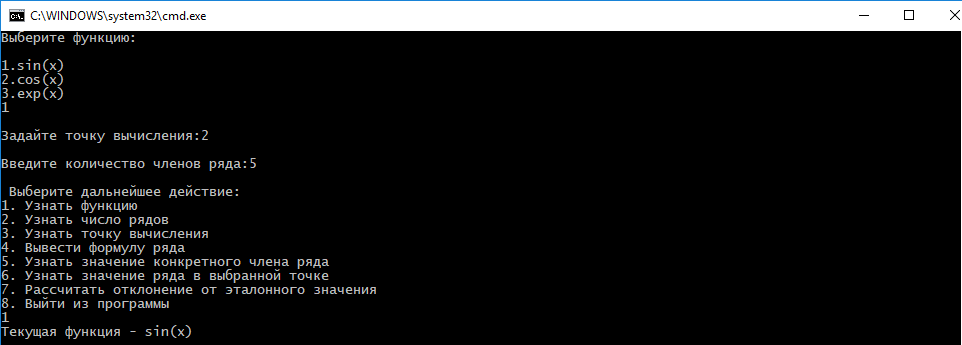


Рис.3. Действие первого пункта программы.

Нажав «4», пользователь увидит формулу Тейлора для введенной функции с введенным количеством членов ряда.

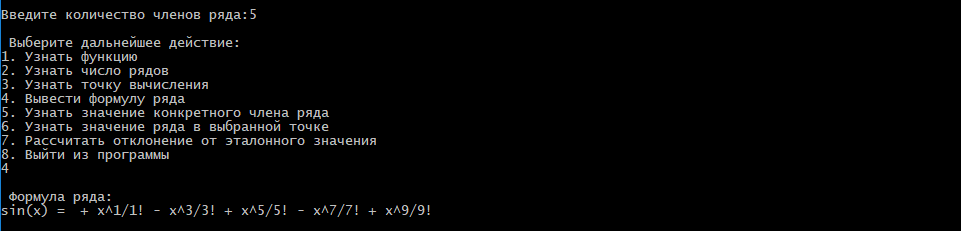


Рис.4. Действие четвертого пункта программы.

Чтобы узнать значение конкретного члена ряда, пользователю предстоит нажать «5».

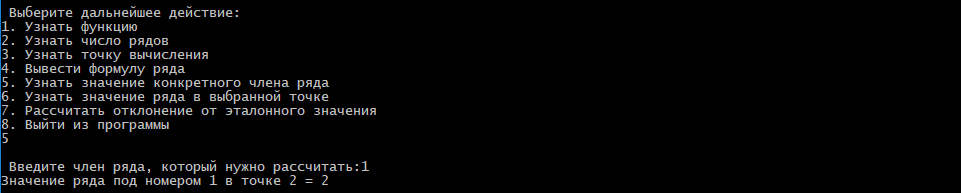


Рис.5. Действие пятого пункта программы.

Чтобы узнать непосредственно само значение ряда, следует нажать «6».

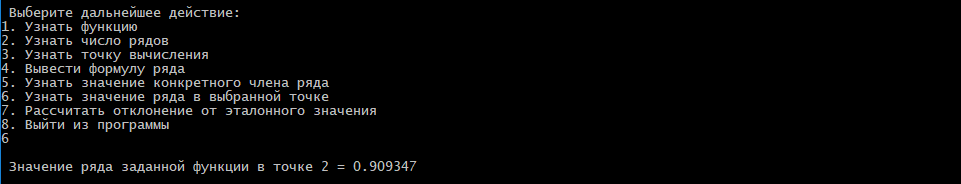


Рис.6. Действие шестого пункта программы.

И, наконец, нажав «7», пользователю будет рассчитано отклонение от эталонного значения.

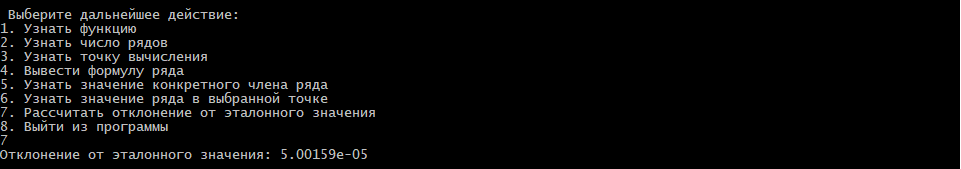


Рис.7. Действие седьмого пункта программы.

# Руководство программиста

В программе реализован один класс под названием «Формулы Тейлора». В этом классе содержится три приватных поля, а именно: переменная типа int, отвечающая за номер выбранной функции, переменная типа int, отвечающая за вычисление функции в произвольно выбранной точке и переменная типа int, отвечающая за количество членов ряда. Помимо этого, в классе имеются следующие методы:

1. Все необходимые служебные методы: конструктор, деструктор, конструктор копирования оператор присваивания,
2. Методы, которые возвращают переменные, отвечающие за номер функции, точку вычисления и количество членов ряда соответственно,
3. Метод, задающие номер функции, количество членов ряда и точку вычисления соответственно,
4. Дружественный метод, печатающий на экран формулу Тейлора для выбранной функции,
5. Метод, возвращающий значение заданного члена ряда,
6. Метод, возвращающий значение ряда в выбранной точке,
7. Метод, возвращающий отклонение от эталонного значения.

## **Описание структуры программы**

Программа построена на использовании интуитивно понятного пользователю меню.

В основной части главной функции программы main создается два объекта типа «Формулы Тейлора», проверяется корректность оператора присваивания, реализованного в классе. Главная функция представляет цикл while, в котором содержится оператор множественного выбора swich. В зависимости от выбора пользователем действия, вызывается тот или иной метод класса «Формулы Тейлора».

## **Описание алгоритмов**

1. Методы, которые возвращают переменные, отвечающие за номер функции, точку вычисления и количество членов ряда соответственно.

Все данные методы реализованы как функции, возвращающие переменную типа int. Они ничего не принимают и возвращают соответствующее поле класса.

1. Метод, задающие номер функции, количество членов ряда и точку вычисления соответственно.

Все данные методы реализованы схожим образом. Они принимают переменную типа int и её значение записывают на место соответствующего поля класса «Формулы Тейлора».

1. Дружественный метод, печатающий на экран формулу Тейлора для выбранной функции.

Метод реализован как дружественная функция типа void. Он принимает один параметр, а именно: ссылку на объект класса «Формулы Тейлора». В зависимости от переданного значения оператор множественного выбора печатает на консоль ту или иную функцию и ее формулу ряда.

1. Метод, возвращающий значение заданного члена ряда

Метод реализован как функция, которая возвращает тип данных double. Он принимает переменную, в которой хранится номер текущего ряда и возвращает значение ряда по его номеру.

1. Метод, возвращающий значение ряда в выбранной точке.

Метод возвращает переменную типа double, в которой записано значение ряда суммирования для выбранной пользователем функции.

1. Метод, возвращающий отклонение от эталонного значения.

Метод реализован как функция, возвращающая переменную типа double. В ней хранится разность по модулю между эталонным значением, и значением ряда Тейлора для текущей функции, полученного из предыдущего метода.

**Лабораторная работа №4. Песенник.**

# Постановка задачи

Требуется разработать класс Песенник.

Класс должен хранить информацию о песенных композициях. Каждая песня описывается следующими данными: название, поэт (автор стихов), композитор (автор музыки), исполнитель, название альбома (если входит в какой-то альбом), дата выпуска (день, месяц, год). Песни хранятся упорядоченно по названию. Данные вводятся на русском языке.

Класс должен содержать необходимые служебные методы.

Класс должен предоставлять следующие операции:

1) добавить песню,

2) изменить данные выбранной песни,

3) найти песню по названию и исполнителю,

4) выдать все песни заданного поэта,

5) выдать все песни заданного композитора,

6) выдать все песни заданного исполнителя,

7) узнать текущее число песен,

8) удалить песню,

9) сохранить песенник в файл,

10) считать песенник из файла.

# Руководство пользователя

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++.

На первом этапе работы программы пользователю необходимо выбрать из одиннадцати вариантов то, что умеет сделать программа, а именно:

1. Добавить песню,
2. Вывести все песни на экран,
3. Изменить выбранную песню,
4. Найти песню по названию и исполнителю,
5. Выдать все песни заданного поэта,
6. Выдать все песни заданного композитора,
7. Выдать все песни заданного исполнителя,
8. Узнать количество песен,
9. Удалить выбранную песню,
10. Записать песенник в файл,
11. Считать песенник из файла.



Рис.1. Старт программы

Если пользователь желает добавить песню в песенник, то ему необходимо нажать «1». После этого программа попросить все необходимые данные для музыки, чтобы записать ее в песенник. Добавлять песни можно неограниченное количество раз. Самостоятельно добавим несколько песен.

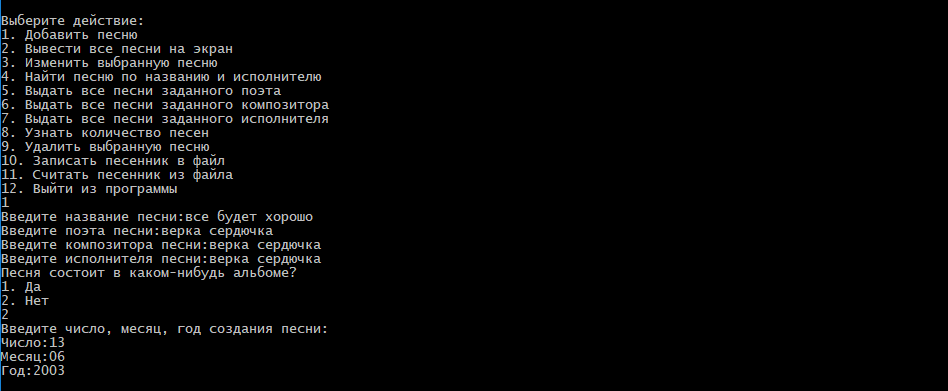


Рис.2. Добавление песни

Нажав «2», пользователь увидит то, какие песни сохранены в системы (какие песни о сохранил)

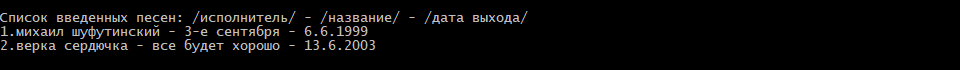


Рис.3. Список введенных песен.

Попробуем изменить конкретную выбранную песню из текущего песенника, нажав на «3».

Программа запрашивает все необходимые данные перед тем как изменить песню.

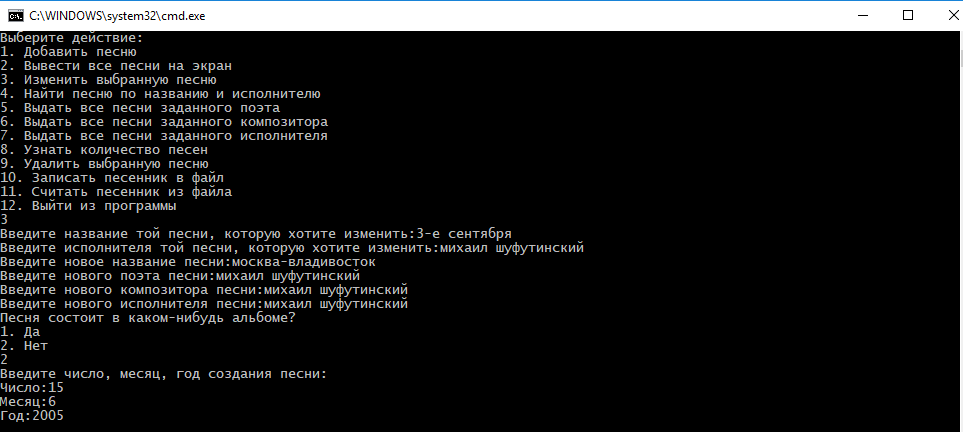
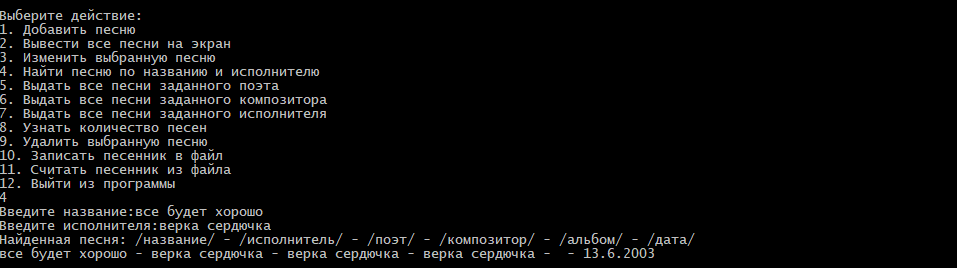
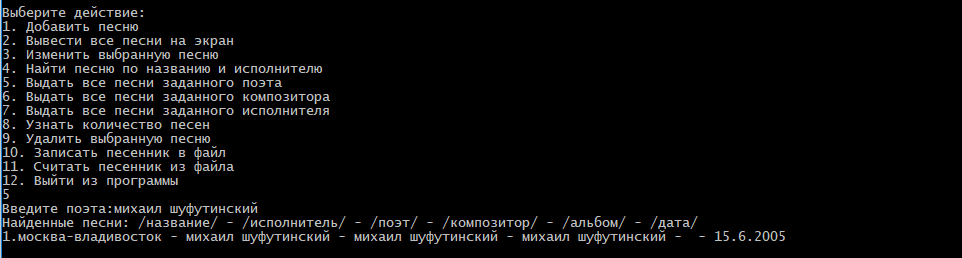


Рис.4. Изменение песни.

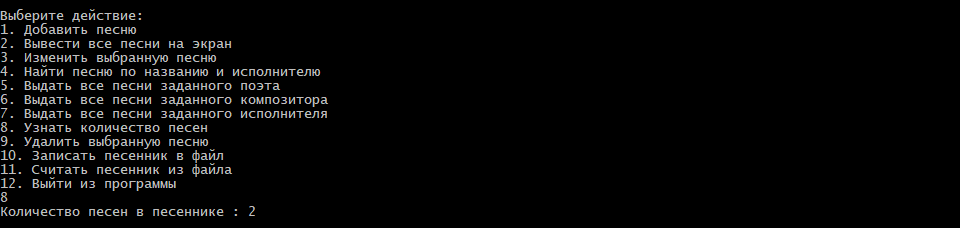
Нажав «4», в песеннике можно найти песню по названию и исполнителю.

Рис. 5. Действие пятого действия программы

Если пользователь желает найти песни по поэту, композитору или исполнителю, необходимо использовать «5», «6» и «7» соответственно.

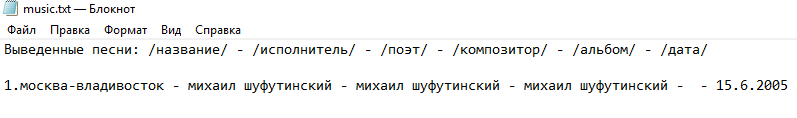
Рис. 6. Поиск песни по поэту.

Если нажать на «8», можно узнать количество песен в песеннике.

Рис.7. Количество песен в песеннике

Если пользователь желает удалить выбранную песню из песенника, необходимо нажать «9».

Песенник также можно записать файл и считать из файла. Для этого необходимо воспользоваться клавишами «10» и «11».

Рис. 8. Сохранение песенника в файл.

# Руководство программиста

В программе реализованы два класса с названиями «Песни», «Песенник» и структура, отвечающая за дату песни. Класс песня имеет несколько приватных полей, а именно: переменные типа string, отвечающие за название песни, название альбома, в котором содержится песня, имя поэта, композитора, исполнителя. Помимо этого, класс «Песня имеет следующие методы:

1. Служебные методы (конструктор, деструктор, конструктор копирования, оператор присваивания, оператор сравнения, оператор меньше),
2. Методы, возвращающие соответственно название песни, название альбома песни, имя исполнителя, поэта, композитора.

Кроме того, в программе реализован другой класс «Песенник», в котором одно приватное поле: вектор типа данных «Песня». Там хранятся песни, которые содержатся в песеннике. Классу свойственны следующие методы:

1. Служебные методы (конструктор, деструктор),
2. Метод, возвращающий по ссылке приватный вектор,
3. Методы, которые возвращают вектор типа данных «Песня» по исполнителю, поэту и композитору,
4. Метод, возвращающий переменную, которая хранит в себе информацию о количестве песен в песеннике,
5. Метод, удаляющий песню из вектора,
6. Метод, записывающий песенник в файл,
7. Метод, считывающий песенник из файла.

## **Описание структуры программы**

Программа построена на использовании интуитивно понятного пользователю меню.

В основная части главной функции программы main создается объект типа «Песенник». Главная функция представляет собой цикл while, в котором расположен оператор множественного выбора swich. В зависимости от выбора пользователем действия, вызывается тот или иной метод класса «Песенник».

## **Описание алгоритмов**

Опишем алгоритмы реализации методов класса «Песня». Методы, которые возвращают требуемое значение из поля класса, реализованы как функции типа «строка» и «дата песни» (для альбома). Все они не принимают никаких параметров и возвращают соответствующее значение переменной из поля класса «Песня».

Что касается класса «Песенник», то в нем реализованы такие методы, как:

1. Метод, возвращающий по ссылке приватный вектор.

Метод реализован как функция, возвращающая вектор из приватного поля класса «Песенник».

1. Методы, которые возвращают вектор типа данных «Песня» по исполнителю, поэту и композитору.

Данные методы однотипные и реализованы как функции, возвращающие переменную типа данных «строка». В одном случае переменная отвечает за имя исполнителя, в другом за имя поэта, в третьем за имя композитора.

1. Метод, возвращающий переменную, которая хранит в себе информацию о количестве песен в песеннике.

Каждый раз, когда вызывается этот метод возвращается размер вектора типа данных «Песня», хранящегося в приватном поле самого класса «Песенник».

1. Метод, удаляющий песню из вектора.

Данный метод реализован как функция типа void, принимающая ссылку на объект типа «Песня». С помощью цикла for песня-аргумент ищется в приватном векторе. Как только она нашлась, она удаляется с помощью одно из методов стандартного класса «вектор».

1. Метод, записывающий песенник в файл.

С помощью этого метода песенник записывается в txt файл. Реализован как функция void без каких-либо параметров.

1. Метод, считывающий песенник из файла.

Данный метод реализован как функция типа void, без каких-либо параметров. Основная идея: цикл while до тех пор, пока не кончатся данные в том файле, из которого считываются файлы. Одна итерация цикла анализирует строку и каждую её часть запоминает в соответствующие переменные.

**Лабораторная работа №5. Билетная касса.**

# Постановка задачи

Разработать классы Билетная касса и Кинотеатр.

Класс Билетная касса должен имитировать работу кассы по продаже билетов на киносеансы в многозальном кинотеатре. Считать, что продажа билетов проводится на сеансы в пределах трех дней от текущей даты. Каждый сеанс описывается датой, временем начала сеанса, названием фильма, номером зала, стоимостью билета в зависимости от зоны (VIP и обычная). Для упрощения считать, что покупатель указывает тип зоны и требуемое число билетов, а места выделяются кассой автоматически (при наличии свободных). Зрительные места в каждом зале описываются номером ряда и номером в ряду. Для упрощения считать, что число мест во всех рядах в одном зале одинаково. Продажа билетов на сеанс прекращается через 10 минут после начала сеанса.

Информация о всех сеансах на ближайшие 30 дней проката хранится в классе Кинотеатр. Для каждого зала установлена базовая стоимость билетов (на дневные сеансы – от 12.00 до 18.00). Стоимость билетов на утренние сеансы (до 12.00) составляет 75% от базовой, стоимость билетов на вечерние сеансы (после 18.00) – 150% от базовой. Информация о зрительных местах (свободно/занято) в каждом зале на каждом сеансе также хранится в классе Кинотеатр.

Класс Билетная касса должен предоставлять следующие операции: 1) принять данные покупателя: дату, время сеанса, название фильма, номер зала, тип зоны, число мест, 2) проверить наличие требуемого количества свободных мест в требуемой зоне, 3) зарезервировать требуемое количество мест, 4) рассчитать общую стоимость билетов, 5) отменить заказ билетов, 6) сформировать билеты (каждый билет включает: дату, время сеанса, название фильма, номер зала, номер ряда, номер места в ряду).

Класс Кинотеатр должен использоваться для поддержки работы класса Билетная касса и может быть разработан в минимально-необходимом варианте.

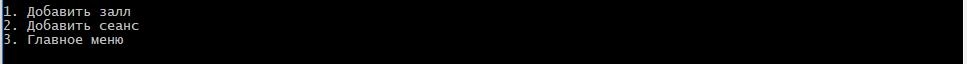
# Руководство пользователя

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++.

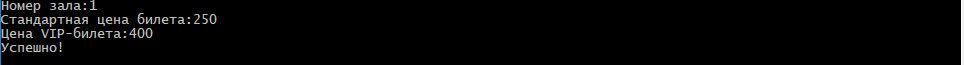
На первом этапе работы программы пользователю необходимо структуру, с которой будет работать в дальнейшем.

Рис.1. Старт программы.

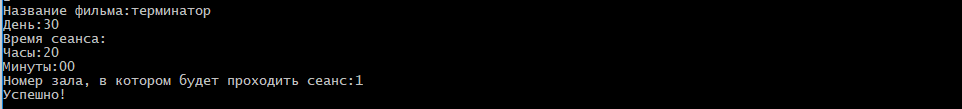
Рассмотрим сначала структуру кинотеатра. В ней можно добавить кинотеатр и добавить новый сеанс.

Рис.2. Предлагаемые действия программы структуры кинотеатра.

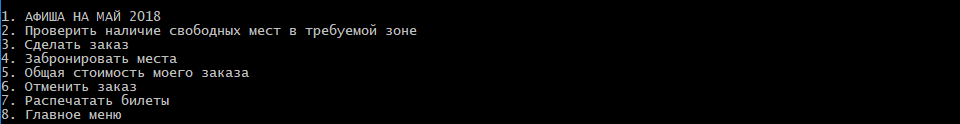
Добавляя зал кинотеатра, пользователю будет необходимо ввести данные, а именно: номер зала, стандартная цена билета, цена VIP-билета.

Рис.3. Добавление зала кинотеатра.

После того, как будет добавлен зал, у пользователя будет возможность добавить некоторое количество сеансов, которое будет проходить в этом зале. Также необходимо будет ввести те данные, которые запросит программа, а именно: название фильма, дата его показа и номер зала, в котором он будет проходить.

Рис. 4. Добавление сеанса кинотеатра.

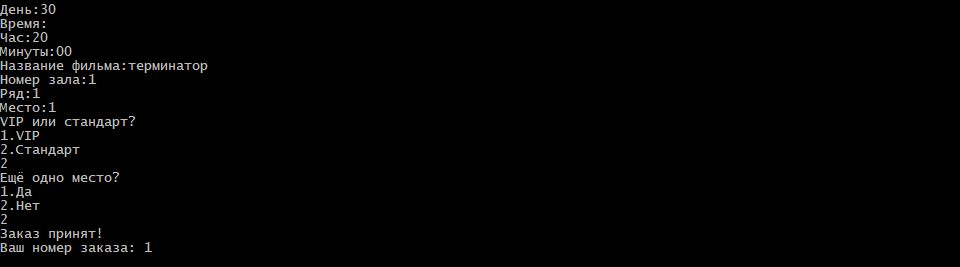
Поле того, как созданы зал и сеансы, можно непосредственно переходить к структуре заказчика и принимать различные заказы.

Рис.5. Структура заказчика.

Чтобы узнать все сеансы на май 2018, пользователю необходимо нажать «1». Будет показано то количество фильмов, которое пользователь добавил в структуре кинотеатра.

Рис.6. Афиша.

Кроме того, пользователю предоставляется возможность работать с заказами. Ему доступна проверка на наличие свободных мест в требуемой зоне, заказ, бронировка мест. При заказе пользователю необходимо ввести следующие данные: день, время, название фильма, ряд, место, билет (VIP или стандарт). Перед заказом автоматически будет проверено наличие свободных мест. Если места свободны, программа одобрит предложение, иначе – сообщит о том, что места заняты. Если заказ одобрен, то пользователю сообщается номер заказа, по которому дальше ориентируется программа.

Рис.7 . Заказ.

Как только программа одобрила заказ, пользователю предоставляется возможность сразу их забронировать. Для этого необходимо ввести номер заказа, полученного в результате самого заказа.

Рис.8. Бронирование билетов.

После всех действий, описанных выше, программа предоставляет возможность вывести на экран общую стоимость всех билетов, распечатать билеты или удалить заказ. Все это также делается по номеру заказа пользователя. Цена билета зависит от времени, в которое будет проходить сеанс.

Рис.9. Информация о цене билета

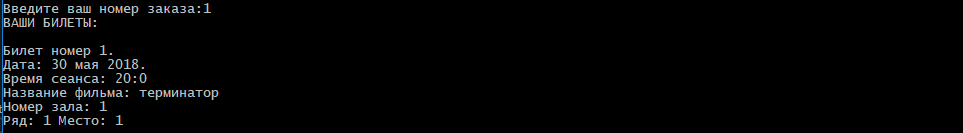
Рис 10. Билеты

Рис 11. Удаление заказа

# Руководство программиста

В программе содержатся три класса «Сеанс», «Кинотеатр» и «Билетная касса». Кроме того, имеются следующие структуры: структура времени начала сеанса, структура зала, структура места, структура заказа и структура сеанса.

Класс «Сеанс» имеет такие приватные поля, как: переменная типа string, отвечающую за название фильма, переменную типа int, отвечающую за день, в который будет проходить сеанс, переменную типа данных структура «time», отвечающую за время сеанса, переменную типа int, отвечающую за номер зала, в котором будет проходить сеанс и вектор векторов типа bool, который хранит информацию о том, заняты ли конкретные места или нет. Помимо этого, класс содержит две статические переменные, она из которых отвечает за количество рядов, другая за количество мест в одном ряду. Кроме того, класс содержит следующие методы:

1. Служебные методы (конструктор, деструктор, конструктор копирования, оператор присваивания, оператор сравнения)
2. Метод, который изменяет данные сеанса,
3. Метод, проверяющий место на то, занято ли оно или нет,
4. Метод, бронирующий места на текущий сеанс,
5. Метод, который отменяет бронирование на текущий сеанс,
6. Методы, возвращающие соответствующие поля из класса.

Класс «Кинотеатр» имеет такие приватные поля, как: вектор векторов типа данных класс «сеанс», вектор типа данных структура «зал». Кроме того, класс имеет следующие необходимые методы:

1. Конструктор,
2. Методы, возвращающие соответствующие поля из класса,
3. Метод, возвращающий сеанс по дню, времени и залу,
4. Метод, возвращающий вектор объектов класса «сеанс» со всеми, реализованными в программе, сеансами,
5. Метод, добавляющий сеанс в конкретный день,
6. Метод, добавляющий зал.

Класс «Билетная касса» содержит в себе объект класса «Кинотеатр», чтобы напрямую иметь доступ к полям этого класса. Кроме того, он имеет вектор типа структура «заказ». В этот вектор поочередно складываются заказы от пользователей. Помимо этого, класс имеет следующие необходимые методы:

1. Конструктор,
2. Метод, отвечающий за заказ от пользователя,
3. Метод, проверяющие наличие мест в той или иной зоне.
4. Метод, бронирующий места,
5. Метод, рассчитывающий общую стоимость заказов,
6. Метод, отвечающий за удаление заказа,
7. Метод, формирующий билеты.

## **Описание структуры программы**

Программа построена на использовании интуитивно понятного пользователю меню.

В основной части главной функции программы main создается объект типа «Кинотеатр» и объект типа «Билетная касса». Главная функция представляет цикл while, в котором содержится оператор множественного выбора swich. В зависимости от выбора пользователем действия, вызывается тот или иной метод класса «Билетная касса».

## **Описание алгоритмов**

**Опишем алгоритмы класса «Сеанс».**

1. Метод, который изменяет данные сеанса.

Метод реализован как функция типа void. Принимает значения текущих полей и поочередно присваивает их переменным, хранящимся в самих полях.

1. Метод, проверяющий место на то, занято ли оно или нет

Метод реализован как функция типа bool, которая принимает две переменные, отвечающие за место в зале. Возвращает true, если это место свободно, иначе – false.

1. Метод, бронирующий места на текущий сеанс и метод, который отменяет бронирование на текущий сеанс

Методы реализованы как функции типа void, принимающая вектор всех мест. Если пользователь хочет забронировать место – советующему значению в приватном векторе присваивается false, иначе – true.

1. Методы, возвращающие соответствующие поля из класса

Методы реализованы как функции, возвращающие соответствующие значения приватных переменных.

**Опишем алгоритмы класса «Кинотеатр».**

1. Методы, возвращающие соответствующие поля из класса.

Данные методы однотипные, реализованы как функции, по ссылке возвращающие соответствующие вектора.

1. Метод, возвращающий сеанс по дню, времени и залу.

Метод реализован как функция, возвращающая объект класса «сеанс». С помощью цикла for сравниваются соответствующие сеансы и подходящий возвращается.

1. Метод, возвращающий вектор объектов класса «сеанс» со всеми, реализованными в программе, сеансами.

Метод реализован как функция, возвращающая вектор объектов класса «сеанс». В самом методе создается этот вектор, куда с помощью цикла for сохраняются все сеансы.

1. Метод, добавляющий сеанс в конкретный день и метод, добавляющий зал.

Методы реализованы как функции типа void, не принимающие параметров. С помощью методов стандартного класса вектор сохраняется сеанс, который будет проходить в тот или иной день или зал со своими параметрами.

**Опишем алгоритмы класса «Билетная касса».**

1. Метод, отвечающий за заказ от пользователя.

Данный метод реализован как функция типа int, принимающая переменную типа структура «заказ». С помощью методов стандартного класса вектор сохраняется заказ, сделанный тем или иным пользователем. Возвращает метод номер текущего заказа.

1. Метод, проверяющие наличие мест в той или иной зоне.

Данный метод реализован как функция типа bool. Она принимает два параметра: вектор всех мест, которые нужно проверить и сеанс. Метод с помощью цикла for ищет подходящий сеанс и для текущего сеанса вызывает метод проверки места, реализованного в самом классе «сеанс». Если места свободны, возвращается true, иначе – false.

1. Метод, бронирующий места.

Метод реализован как функция типа bool, принимающая номер заказа пользователя. Если места успешно забронированы, то метод возвращает true, иначе – false.

1. Метод, рассчитывающий общую стоимость заказов.

Метод реализован как функция типа double, принимающая номер заказа пользователя. В нем создается сама переменная и с помощью цикла for постепенно записывается значение.

1. Метод, отвечающий за удаление заказа.

Данный метод реализован как функция типа void, принимающая номер заказа от пользователя. С помощью метода стандартного класса вектор удаляется заказ, сделанный тем или иным пользователем.

1. Метод, формирующий билеты.

Данный метод реализован как фикция, возвращающая вектор типа структура «билет», принимает номер заказа от пользователя и с помощью цикла for поочередно сохраняет в вектор билеты по месту.

**Лабораторная работа №6. Игра «Морской бой».**

# Постановка задачи

Требуется разработать систему классов и реализовать с ее помощью игру Морской бой.

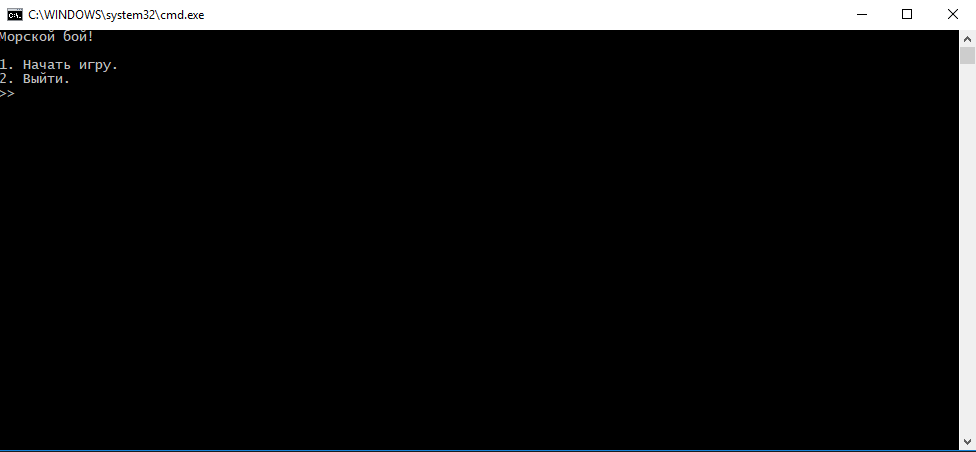
**Требования (правила):**

1. Играют два игрока (человек и компьютер).
2. У каждого игрока два поля 10x10 клеток. В левом поле игрок расставляет свои корабли. В правом игрок пытается потопить чужие корабли.
3. У каждого игрока имеются 4 «однопалубных» (из одной клетки) корабля, 3 «двухпалубных», 2 «трехпалубных» и 1 «четырехпалубный» корабль.
4. Многопалубные корабли могут располагаться только по горизонтали или вертикали.
5. Корабли не могут располагаться в соседних клетках. Соседними для каждой клетки считаются 8 окружающих ее клеток.
6. Игра состоит из поочередных ходов игроков.
7. Первый ход выполняет человек.
8. Каждый ход состоит из следующих действий:
   1. Игрок, выполняющий ход, «называет» выбранную клетку (координаты).
   2. Соперник проверяет «попадание» на своей доске. Если в названной клетке расположен корабль, соперник оглашает попадание, иначе промах.
   3. Игрок, выполняющий ход, ставит на своей правой доске по названным координатам отметку о результатах хода.
   4. Если игрок, выполняющий ход, попал в корабль, ход остается у него, иначе переходит к сопернику.
9. Выигрывает тот игрок, кто первым потопит все корабли противника.

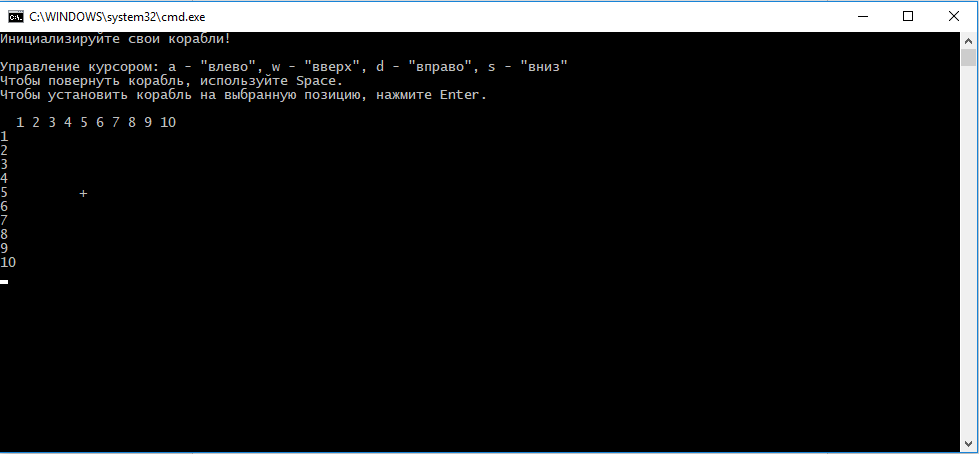
# Руководство пользователя

Данная программа написана в среде разработки Microsoft Visual Studio 2017 на языке С++.

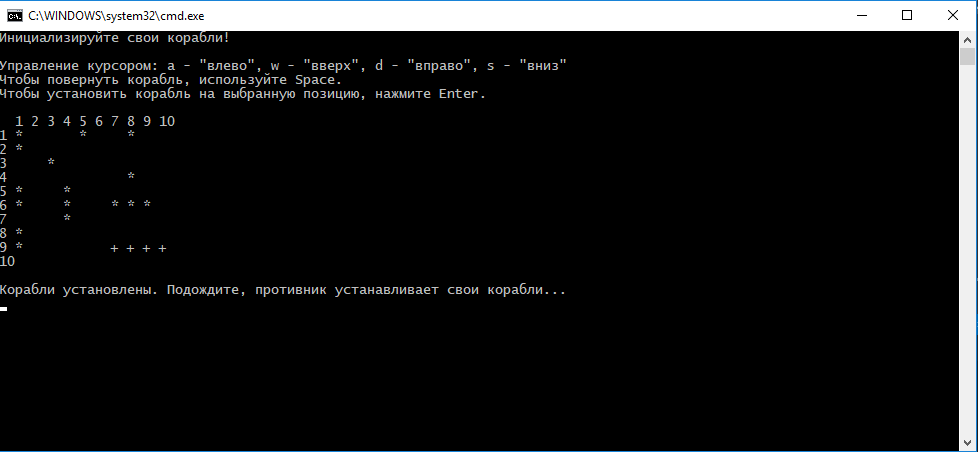
На первом этапе работы программы выводится меню игры.

Рис.1. Старт программы.

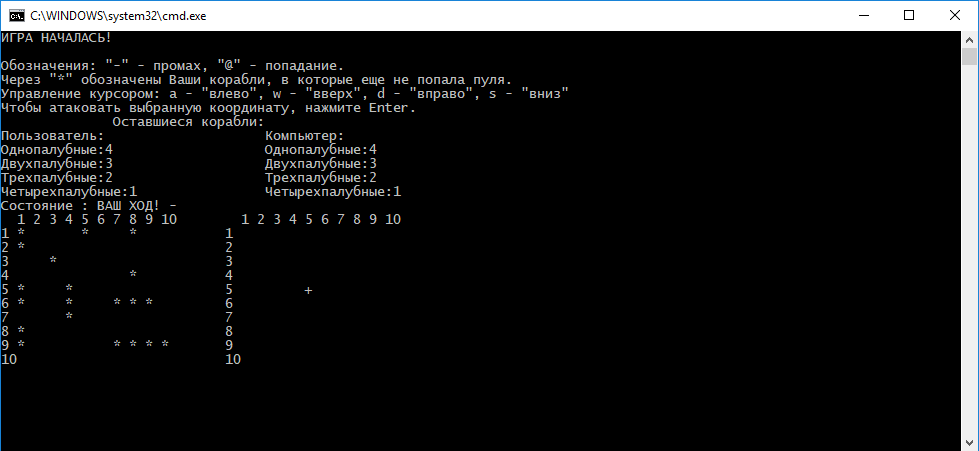
Начав игру, пользователю необходимо расставить свои корабли.

Рис.2. Расставление кораблей.

После того, как пользователь расставит корабли, игрок-компьютер начнет расставлять свои корабли.

Рис.3. Сообщение о расстановки кораблей компьютера.

После того, как компьютер расставит свои корабли, игра начнется.

Рис. 4. Начало игры.

Первым ходит игрок-пользователь, стреляя по полю противника. Если промах – ход переходит к компьютеру. Если попадание, пользователь стреляет еще раз. Попадая, игрок-компьютер пытается сразу же добить корабль, в который попал изначально.

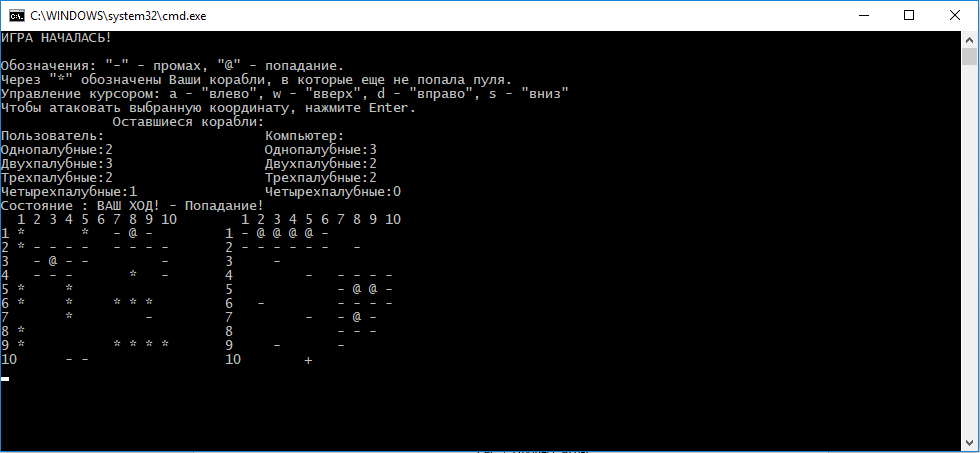


Рис.5. Процесс игры.

После окончания игры, программа сообщит, кто одержал победу.

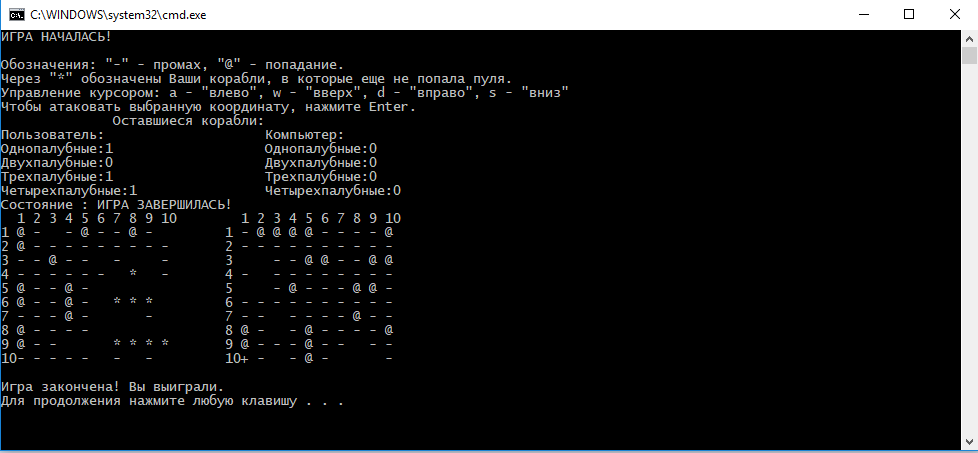


Рис.6. Окончание игры.

# Руководство программиста

В программе содержатся класс «Игрок морского боя», от которого наследуются два класса «Игрок-пользователь» и «Игрок-компьютер». Кроме того, имеются следующие структуры: структуры однопалубного, двухпалубного, трехпалубного и четырехпалубного кораблей. Также реализованы четыре вспомогательные функции: функция, избавляющая от мерцаний, функция, проверяющая на корректность координаты, и две функции вывода на экран одного и того же.

Структуры однопалубного, двухпалубного, трехпалубного и четырехпалубного кораблей реализованы сходим образом, имеют переменные типа int, отвечающие за каждую из координат и переменную типа bool, отвечающую за положение корабля (вертикальное или горизонтальное) (за исключением однопалубного корабля). Кроме того, структуры имеют следующие методы:

1. Метод, изменяющий координату корабля,
2. Метод, присваивающий промахи окружению того или иного корабля,

**Класс «Игрок морского боя»**

Класс «Игрок морского боя» содержит следующие приватные переменные (реализованные в секции protected): вектор из однопалубных кораблей, вектор из трехпалубных кораблей, вектор из двухпалубных кораблей и переменную типа структура «четырехпалубный корабль». Кроме того, имеется вектор векторов типа bool, который фактически является своим фронтом и аналогичный вектор, являющийся фронтом противника. Помимо этого, в этой секции имеется конструктор, ибо создавать объект такого типа категорически запрещено.

Что касается секции public, то здесь имеются следующие статические переменные:

1. Статические копии векторов однопалубных, двухпалубных, трехпалубных кораблей и статическая копия четырехпалубного корабля (используются для того, чтобы корабли были видны по всему наследованию),
2. Статические переменные типа int, отвечающие за количество неубитых кораблей у пользователя и компьютера.

Реализованы следующие методы:

1. Методы, возвращающие векторы векторов типа bool (фронт противника и свой фронт),
2. Метод, задающий значение фронту противника,
3. Методы, проверяющий корабль на правильность расположения (при инициализации).

**Класс «Игрок-пользователь»**

Класс «Игрок-пользователь» не содержит переменных, объявленных в секции private. Класс имеет следующие статические переменные:

1. Статическая переменная, считающая корабли,
2. Статическая переменная, отвечающая за положение корабля (вертикально, горизонтально).

Класс «Игрок-пользователь» содержит следующие методы:

1. Методы, возвращающие однопалубный, двухпалубный, трехпалубный корабль по координате у компьютера,
2. Метод, который узнает, в какой корабль попала пуля у компьютера,
3. Метод, инициализирующий корабли пользователя,
4. Метод, рисующий вспомогательную карту врага и свой фронт,
5. Основной метод, отвечающий за атаку по компьютеру,
6. Вспомогательный метод, отвечающий за стрельбу по компьютеру.
7. Метод, проверяющий «конец игры»

**Класс «Игрок-компьютер»**

Класс «Игрок-пользователь» не содержит переменных, объявленных в секции private. Класс имеет следующие статические переменные:

1. Вспомогательные статические переменная для атаки по пользователю. Именно с помощью них добивается корабль пользователя, если игрок-компьютер изначально в него попал,
2. Статическая переменная, помогающая правильно поставить корабли компьютеру.

Класс «Игрок-компьютер» содержит следующие методы:

1. Методы, возвращающие однопалубный, двухпалубный, трехпалубный корабль по координате у пользователя,
2. Метод, который узнает, в какой корабль попала пуля у пользователя,
3. Метод, инициализирующий корабли компьютера,
4. Метод, возвращающий координаты корабля по одной из них у пользователя,
5. Основной метод, отвечающий за атаку по пользователю,
6. Вспомогательный метод, отвечающий за стрельбу у пользователя.
7. Метод, проверяющий «конец игры»

## **Описание структуры программы**

Программа построена на использовании интуитивно понятного пользователю меню.

В основной части главной функции программы main создается объект типа «Игрок-компьютер» и «Игрок-пользователь». Для каждого объекта вызывается своего рода метод, который отвечает за инициализацию кораблей. Другая часть функции представляет цикл while, в котором поочередно вызывается своего рода метод, отвечающий за стрельбу.

## **Описание алгоритмов**

**Опишем алгоритмы методов структур кораблей.**

1. Метод, изменяющий координату корабля.

Метод реализован как функция типа void, которая принимает координату и в зависимости от нее присваивает значения другим координатам корабля. Метод реализован с целью исправления координат корабля в случае некорректного рандома компьютера при инициализации кораблей.

1. Метод, присваивающий промахи окружению того или иного корабля.

Когда корабль полностью уничтожен, необходимо окружению корабля в фронте присвоить «промахи». Метод реализован именно с этой целью и с помощью двойного цикла for это действие выполняется.

**Опишем алгоритмы класса «Игрок морского боя».**

1. Методы, возвращающие векторы векторов типа bool (фронт противника и свой фронт)

Метод реализован как функция, возвращающая по ссылке вектор векторов типа bool. С точки зрения проверки на попадание этот метод необходим.

1. Метод, задающий значение фронту противника.

Метод реализован как функция, тип которой void, принимает в параметре вектор векторов типа bool и присваивает его соответствующему полю класса.

1. Методы, проверяющий корабль на правильность расположения (при инициализации).

Метод реализован как функция типа bool. Если корабль убит, вызывается этот метод, присваивающий «промахи» окружению корабля.

**Опишем алгоритмы класса «Игрок-пользователь и игрок-компьютер».**

Методы двух классов схожи между собой, однако обращаются они к разным векторам.

1. Методы, возвращающие однопалубный, двухпалубный, трехпалубный корабль по координате у компьютера/пользователя.

Данные методы реализованы как функции, возвращающие соответственный вектор из однопалубных, двупалубных, трехпалубных кораблей, принимающие координату. С помощью цикла for возвращают тот корабль у компьютера/пользователя, который требуется.

1. Метод, который узнает, в какой корабль попала пуля у компьютера/пользователя.

Метод реализован, как функция, возвращающая переменную int и принимающая координату. Метод может вернуть четыре значения: 1 – пуля попала в однопалубный, 2 – пуля попала в двухпалубный, и т.д. С помощью цикла for проверяются соответствующие вектора и возвращается нужное значение.

1. Метод, инициализирующий корабли компьютера.

Метод реализован как функция типа void, не принимающая параметров. С помощью нее корректно инициализируются корабли пользователя. Координаты инициализируются с помощью функции rand. Если некорректно инициализировалась координата, вызывается метод, изменяющий координаты у соответствующего корабля

1. Метод, инициализирующий корабли пользователя

Данный метод реализован как функция типа void, принимающая координату по ссылке. Пользователь расставляет корабли и в зависимости от этого корабли принимают некоторые координаты.

1. Метод, отвечающий за атаку по компьютеру/пользователю

Методы реализованы как функции, возвращающие переменную int. Метод может вернуть 3 значения, а именно: 0 – мимо, 1 – попал, 2 – убил. Метод обращается к своему вспомогательному и анализирует вектор векторов типа int, где хранятся попадания пользователя/компьютера.

# Заключение

Второй семестр дал мне колоссальный опыт по работе с классами и объектно-ориентированного программирования в целом. Я убедился в том, что объектно-ориентированное программирование является основой всей индустрии прикладного программирования благодаря выигрышу в конкурентной борьбе с альтернативными технологиями программирования.

# Литература

1. Бьёрн Страуструп. Язык программирования C++ — 3-е изд. — СПб.; М.: Невский диалект — Бином, 1999. — 991 с.

2. Герберт Шилдт. Полный справочник по C++ — 4-е изд. — М.: Вильямс, 2011. — 800 c.

# 

# Приложение

#include <iostream>

#include <Windows.h>

#include <vector>

#include <conio.h>

#include <cstdlib>

#include <string>

#include <ctime>

using namespace std;

void setcur(int x, int y) //вспомогательная функция для прекращения мерцаний.

{

COORD coord;

coord.X = x;

coord.Y = y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), coord);

};

struct ship\_1

{

int x1\_ship1;

int y1\_ship1;

ship\_1(int a = 0, int b = 0): x1\_ship1(a), y1\_ship1(b) {}

void change(int a, int b)

{

x1\_ship1 = a;

y1\_ship1 = b;

}

void transformation1(vector<vector<int>> & v)

{

int x\_copy = x1\_ship1 - 1;

int y\_copy = y1\_ship1 - 1;

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

{

if (x\_copy + j <= 0 || x\_copy + j >= 11 || y\_copy + i <= 0 || y\_copy + i >= 11)

continue;

else

v[y\_copy + i - 1][x\_copy + j - 1] = 1;

}

}

};

struct ship\_2

{

int testen\_two = 2;

int x1\_ship2;

int y1\_ship2;

int x2\_ship2;

int y2\_ship2;

bool position2; // =true, если корабль вертикален, =false, если корабль горизонтален.

ship\_2(int a = 0, int b = 0, bool \_position2 = false) : x1\_ship2(a), y1\_ship2(b), position2(\_position2)

{

if (\_position2)

{

x2\_ship2 = a;

y2\_ship2 = b + 1;

}

else

{

x2\_ship2 = a + 1;

y2\_ship2 = b;

}

}

void change(int a, int b)

{

x1\_ship2 = a;

y1\_ship2 = b;

if (position2)

{

x2\_ship2 = a;

y2\_ship2 = b + 1;

}

else

{

x2\_ship2 = a + 1;

y2\_ship2 = b;

}

}

void transformation2(vector<vector<int>> & v)

{

int x\_copy = x1\_ship2 - 1;

int y\_copy = y1\_ship2 - 1;

if (position2)

{

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

for (int j = 0; j < 3; j++)

{

if (x\_copy + j <= 0 || x\_copy + j >= 11 || y\_copy + i <= 0 || y\_copy + i >= 11)

continue;

else

{

if (v[y\_copy + i - 1][x\_copy + j - 1] == 2)

continue;

v[y\_copy + i - 1][x\_copy + j - 1] = 1;

}

}

}

else

{

for (int i = 0; i < 3; i++)

for (int j = 0; j < 4; j++)

{

if (x\_copy + j <= 0 || x\_copy + j >= 11 || y\_copy + i <= 0 || y\_copy + i >= 11)

continue;

else

{

if (v[y\_copy + i - 1][x\_copy + j - 1] == 2)

continue;

v[y\_copy + i - 1][x\_copy + j - 1] = 1;

}

}

}

}

};

struct ship\_3

{ int testen\_three = 3;

int x1\_ship3;

int y1\_ship3;

int x2\_ship3;

int y2\_ship3;

int x3\_ship3;

int y3\_ship3;

bool position3; // =true, если корабль вертикален, =false, если корабль горизонтален.

ship\_3(int a = 0, int b = 0, bool \_position3 = false) : x1\_ship3(a), y1\_ship3(b), position3(\_position3)

{

if (\_position3)

{

x2\_ship3 = a;

y2\_ship3 = b + 1;

x3\_ship3 = a;

y3\_ship3 = b + 2;

}

else

{

x2\_ship3 = a + 1;

y2\_ship3 = b;

x3\_ship3 = a + 2;

y3\_ship3 = b;

}

}

void change(int a, int b)

{

x1\_ship3 = a;

y1\_ship3 = b;

if (position3)

{

x2\_ship3 = a;

y2\_ship3 = b + 1;

x3\_ship3 = a;

y3\_ship3 = b + 2;

}

else

{

x2\_ship3 = a + 1;

y2\_ship3 = b;

x3\_ship3 = a + 2;

y3\_ship3 = b;

}

}

void transformation3(vector<vector<int>> & v)

{

int x\_copy = x1\_ship3 - 1;

int y\_copy = y1\_ship3 - 1;

if (position3)

{

for (int i = 0; i < 5; i++)

for (int j = 0; j < 3; j++)

{

if (x\_copy + j <= 0 || x\_copy + j >= 11 || y\_copy + i <= 0 || y\_copy + i >= 11)

continue;

else

{

if (v[y\_copy + i - 1][x\_copy + j - 1] == 2)

continue;

v[y\_copy + i - 1][x\_copy + j - 1] = 1;

}

}

}

else

{

for (int i = 0; i < 3; i++)

for (int j = 0; j < 5; j++)

{

if (x\_copy + j <= 0 || x\_copy + j >= 11 || y\_copy + i <= 0 || y\_copy + i >= 11)

continue;

else

{

if (v[y\_copy + i - 1][x\_copy + j - 1] == 2)

continue;

v[y\_copy + i - 1][x\_copy + j - 1] = 1;

}

}

}

}

};

struct ship\_4

{

int testen\_four = 4;

int x1\_ship4;

int y1\_ship4;

int x2\_ship4;

int y2\_ship4;

int x3\_ship4;

int y3\_ship4;

int x4\_ship4;

int y4\_ship4;

bool position4; // =true, если корабль вертикален, =false, если корабль горизонтален.

ship\_4(int a = 0, int b = 0, bool \_position4 = false) : x1\_ship4(a), y1\_ship4(b), position4(\_position4)

{

if (\_position4)

{

x2\_ship4 = a;

y2\_ship4 = b + 1;

x3\_ship4 = a;

y3\_ship4 = b + 2;

x4\_ship4 = a;

y4\_ship4 = b + 3;

}

else

{

x2\_ship4 = a + 1;

y2\_ship4 = b;

x3\_ship4 = a + 2;

y3\_ship4 = b;

x4\_ship4 = a + 3;

y4\_ship4 = b;

}

}

void change(int a, int b)

{

x1\_ship4 = a;

y1\_ship4 = b;

if (position4)

{

x2\_ship4 = a;

y2\_ship4 = b + 1;

x3\_ship4 = a;

y3\_ship4 = b + 2;

x4\_ship4 = a;

y4\_ship4 = a + 3;

}

else

{

x2\_ship4 = a + 1;

y2\_ship4 = b;

x3\_ship4 = a + 2;

y3\_ship4 = b;

x4\_ship4 = a + 3;

y4\_ship4 = b;

}

}

void transformation4(vector<vector<int>> & v)

{

int x\_copy = x1\_ship4 - 1;

int y\_copy = y1\_ship4 - 1;

if (position4)

{

for (int i = 0; i < 6; i++)

for (int j = 0; j < 3; j++)

{

if (x\_copy + j <= 0 || x\_copy + j >= 11 || y\_copy + i <= 0 || y\_copy + i >= 11)

continue;

else

{

if (v[y\_copy + i - 1][x\_copy + j - 1] == 2)

continue;

v[y\_copy + i - 1][x\_copy + j - 1] = 1;

}

}

}

else

{

for (int i = 0; i < 3; i++)

for (int j = 0; j < 6; j++)

{

if (x\_copy + j <= 0 || x\_copy + j >= 11 || y\_copy + i <= 0 || y\_copy + i >= 11)

continue;

else

{

if (v[y\_copy + i - 1][x\_copy + j - 1] == 2)

continue;

v[y\_copy + i - 1][x\_copy + j - 1] = 1;

}

}

}

}

};

bool correctly(int x, int y) //вспомогательная функция для проверки координат

{

if (x == 0 || y == 0 || x == 11 || y == 11)

return false;

else

return true;

}

class PlayerSeaBattle

{

protected:

vector <ship\_1> one\_deck;

vector <ship\_2> two\_deck;

vector <ship\_3> three\_deck;

ship\_4 four\_deck;

vector<vector<bool>> board; //доска кораблей

vector<vector<bool>> board\_enemy; //доска противника(та, которая запоминает ходы)

PlayerSeaBattle() //конструктор по умолчанию

{

one\_deck = vector<ship\_1>();

two\_deck = vector<ship\_2>();

three\_deck = vector<ship\_3>();

vector<vector<bool>> v(10);//

board = v, board\_enemy = v;

for (int i = 0; i < 10; i++)

for (int j = 0; j < 10; j++)

{

board[i].push\_back(true);

board\_enemy[i].push\_back(true);

}

}

public:

//статические переменные

static vector<ship\_1> one\_deck\_use;

static vector<ship\_2> two\_deck\_use;

static vector<ship\_3> three\_deck\_use;

static ship\_4 four\_deck\_use;

static vector<ship\_1> one\_deck\_pc;

static vector<ship\_2> two\_deck\_pc;

static vector<ship\_3> three\_deck\_pc;

static ship\_4 four\_deck\_pc;

static int number\_1\_pc;

static int number\_2\_pc;

static int number\_3\_pc;

static int number\_4\_pc;

static int number\_1\_use;

static int number\_2\_use;

static int number\_3\_use;

static int number\_4\_use;

vector<vector<bool>> ReturnBoard()

{

return board;

}

vector<vector<bool>> ReturnBoardEmeny()

{

return board\_enemy;

}

void GetBoard(vector<vector<bool>> v)

{

board\_enemy = v;

}

bool CheckShip1(ship\_1 sh)

{

int flag1 = 0;

int x0\_1 = sh.x1\_ship1 - 1;

int y0\_1 = sh.y1\_ship1 - 1;

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

{

if (x0\_1 + j <= 0 || x0\_1 + j >= 11 || y0\_1 + i <= 0 || y0\_1 + i >= 11)

{

flag1++;

continue;

}

else

{

if (board[y0\_1 + i - 1][x0\_1 + j - 1] == true)

flag1++;

}

}

if (flag1 == 9)

return true;

else

return false;

}

bool CheckShip2(ship\_2 sh)

{

int flag2 = 0;

int x0\_2 = sh.x1\_ship2 - 1;

int y0\_2 = sh.y1\_ship2 - 1;

if (sh.position2)

{

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

for (int j = 0; j < 3; j++)

{

if (x0\_2 + j <= 0 || x0\_2 + j >= 11 || y0\_2 + i <= 0 || y0\_2 + i >= 11)

{

flag2++;

continue;

}

else

{

if (board[y0\_2 + i - 1][x0\_2 + j - 1] == true)

flag2++;

}

}

if (flag2 == 12)

return true;

else

return false;

}

else

{

for (int i = 0; i < 3; i++)

for (int j = 0; j < 4; j++)

{

if (x0\_2 + j <= 0 || x0\_2 + j >= 11 || y0\_2 + i <= 0 || y0\_2 + i >= 11)

{

flag2++;

continue;

}

else

{

if (board[y0\_2 + i - 1][x0\_2 + j - 1] == true)

flag2++;

}

}

if (flag2 == 12)

return true;

else

return false;

}

}

bool CheckShip3(ship\_3 sh)

{

int flag3 = 0;

int x0\_3 = sh.x1\_ship3 - 1;

int y0\_3 = sh.y1\_ship3 - 1;

if (sh.position3)

{

for (int i = 0; i < 5; i++)

for (int j = 0; j < 3; j++)

{

if (x0\_3 + j <= 0 || x0\_3 + j >= 11 || y0\_3 + i <= 0 || y0\_3 + i >= 11)

{

flag3++;

continue;

}

else

{

if (board[y0\_3 + i - 1][x0\_3 + j - 1] == true)

flag3++;

}

}

if (flag3 == 15)

return true;

else

return false;

}

else

{

for (int i = 0; i < 3; i++)

for (int j = 0; j < 5; j++)

{

if (x0\_3 + j <= 0 || x0\_3 + j >= 11 || y0\_3 + i <= 0 || y0\_3 + i >= 11)

{

flag3++;

continue;

}

else

{

if (board[y0\_3 + i - 1][x0\_3 + j - 1] == true)

flag3++;

}

}

if (flag3 == 15)

return true;

else

return false;

}

}

bool CheckShip4(ship\_4 sh)

{

int flag4 = 0;

int x0\_4 = sh.x1\_ship4 - 1;

int y0\_4 = sh.y1\_ship4 - 1;

if (sh.position4)

{

for (int i = 0; i < 6; i++)

for (int j = 0; j < 3; j++)

{

if (x0\_4 + j <= 0 || x0\_4 + j >= 11 || y0\_4 + i <= 0 || y0\_4 + i >= 11)

{

flag4++;

continue;

}

else

{

if (board[y0\_4 + i - 1][x0\_4 + j - 1] == true)

flag4++;

}

}

if (flag4 == 18)

return true;

else

return false;

}

else

{

for (int i = 0; i < 3; i++)

for (int j = 0; j < 6; j++)

{

if (x0\_4 + j <= 0 || x0\_4 + j >= 11 || y0\_4 + i <= 0 || y0\_4 + i >= 11)

{

flag4++;

continue;

}

else

{

if (board[y0\_4 + i - 1][x0\_4 + j - 1] == true)

flag4++;

}

}

if (flag4 == 18)

return true;

else

return false;

}

}

};

class PlayerUser : public PlayerSeaBattle

{

public:

static int number\_ship; //переменная, считающая корабли

static bool number\_position;//переменная, отвечающая за положение корабля(вертикально, горизонтально)

PlayerUser() : PlayerSeaBattle() {} //конструктор

ship\_1 & ReturnShipFromPc\_1(int x, int y)//вернуть однопалубный корабль у компьютера

{

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

if (one\_deck\_pc[i].x1\_ship1 == x && one\_deck\_pc[i].y1\_ship1 == y)

return one\_deck\_pc[i];

return one\_deck\_pc[0];

}

ship\_2 & ReturnShipFromPc\_2(int x, int y)//вернуть двухпалубный корабль у компьютера

{

for (int j = 0; j < 3; j++)

if ((two\_deck\_pc[j].x1\_ship2 == x && two\_deck\_pc[j].y1\_ship2 == y) || (two\_deck\_pc[j].x2\_ship2 == x && two\_deck\_pc[j].y2\_ship2 == y))

return two\_deck\_pc[j];

return two\_deck[0];

}

ship\_3 & ReturnShipFromPc\_3(int x, int y)//вернуть трехпалубный корабль у компьютера

{

for (int j = 0; j < 2; j++)

if ((three\_deck\_pc[j].x1\_ship3 == x && three\_deck\_pc[j].y1\_ship3 == y) || (three\_deck\_pc[j].x2\_ship3 == x && three\_deck\_pc[j].y2\_ship3 == y) || (three\_deck\_pc[j].x3\_ship3 == x && three\_deck\_pc[j].y3\_ship3 == y))

return three\_deck\_pc[j];

return three\_deck\_pc[0];

}

int InfShipFromPc(int x, int y) // узнать, в какой корабль у компьютера попала пуля.

{

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

if (one\_deck\_pc[i].x1\_ship1 == x &&one\_deck\_pc[i].y1\_ship1 == y)

return 1;

for (int j = 0; j < 3; j++)

if ((two\_deck\_pc[j].x1\_ship2 == x && two\_deck\_pc[j].y1\_ship2 == y) || (two\_deck\_pc[j].x2\_ship2 == x &&two\_deck\_pc[j].y2\_ship2 == y))

return 2;

for (int j = 0; j < 2; j++)

if ((three\_deck\_pc[j].x1\_ship3 == x && three\_deck\_pc[j].y1\_ship3 == y) || (three\_deck\_pc[j].x2\_ship3 == x && three\_deck\_pc[j].y2\_ship3 == y) || (three\_deck\_pc[j].x3\_ship3 == x && three\_deck\_pc[j].y3\_ship3 == y))

return 3;

return 4;

}

int GetAttackOnPc(int x, int y, vector<vector<int>> & v\_enemy) // атака по компьютеру

{

if (board\_enemy[y - 1][x - 1] == true)

return 0;

else

{

if (InfShipFromPc(x, y) == 1)

{

ReturnShipFromPc\_1(x, y).transformation1(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else if (InfShipFromPc(x, y) == 2)

{

ReturnShipFromPc\_2(x, y).testen\_two--;

if (ReturnShipFromPc\_2(x, y).testen\_two == 0)

{

ReturnShipFromPc\_2(x, y).transformation2(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else

{

v\_enemy[y - 1][x - 1] = 2;

return 1;

}

}

else if (InfShipFromPc(x, y) == 3)

{

ReturnShipFromPc\_3(x, y).testen\_three--;

if (ReturnShipFromPc\_3(x, y).testen\_three == 0)

{

ReturnShipFromPc\_3(x, y).transformation3(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else

{

v\_enemy[y - 1][x - 1] = 2;

return 1;

}

}

else if (InfShipFromPc(x, y) == 4)

{

four\_deck\_pc.testen\_four--;

if (four\_deck\_pc.testen\_four == 0)

{

four\_deck\_pc.transformation4(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else

{

v\_enemy[y - 1][x - 1] = 2;

return 1;

}

}

}

return -1;

}

void InitializeShip(int & x, int & y) //инициализировать корабли пользователя.

{

if (number\_ship < 4)

{

cout << " ";

for (int i = 0; i < 10; i++)

cout << i + 1 << " ";

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 20; k++)

{

if (j == 9)

{

if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

}

cout << endl;

}

}

else if (number\_ship >= 4 && number\_ship < 7)

{

if (!number\_position)

{

cout << " ";

for (int i = 0; i < 10; i++)

cout << i + 1 << " ";

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 20; k++)

{

if (j == 9)

{

if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x + 1 && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x + 1 && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

}

cout << endl;

}

}

else

{

cout << " ";

for (int i = 0; i < 10; i++)

cout << i + 1 << " ";

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 20; k++)

{

if (j == 9)

{

if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y + 1)

cout << "+";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y + 1)

cout << "+";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

}

cout << endl;

}

}

}

else if (number\_ship >= 7 && number\_ship < 9)

{

if (!number\_position)

{

cout << " ";

for (int i = 0; i < 10; i++)

cout << i + 1 << " ";

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 20; k++)

{

if (j == 9)

{

if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x + 1 && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x + 2 && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x + 1 && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x + 2 && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

}

cout << endl;

}

}

else

{

cout << " ";

for (int i = 0; i < 10; i++)

cout << i + 1 << " ";

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 20; k++)

{

if (j == 9)

{

if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y + 1)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y + 2)

cout << "+";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y + 1)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y + 2)

cout << "+";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

}

cout << endl;

}

}

}

else if (number\_ship == 9)

{

if (!number\_position)

{

cout << " ";

for (int i = 0; i < 10; i++)

cout << i + 1 << " ";

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 20; k++)

{

if (j == 9)

{

if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x + 1 && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x + 2 && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x + 3 && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x + 1 && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x + 2 && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x + 3 && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

}

cout << endl;

}

}

else

{

cout << " ";

for (int i = 0; i < 10; i++)

cout << i + 1 << " ";

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 20; k++)

{

if (j == 9)

{

if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y + 1)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y + 2)

cout << "+";

else if (k % 2 == 0 && k / 2 + 1 == x && j + 1 == y + 3)

cout << "+";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y + 1)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y + 2)

cout << "+";

else if (k % 2 == 1 && k / 2 + 1 == x && j + 1 == y + 3)

cout << "+";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else

putchar(' ');

}

}

cout << endl;

}

}

}

switch (\_getch())

{

case 'a':

{

if (x > 1)

x--;

break;

}

case 'w':

{

if (y > 1)

y--;

break;

}

case 'd':

{

if (number\_ship < 4)

{

if (x < 10)

x++;

break;

}

else if (number\_ship >= 4 && number\_ship < 7)

{

if (!number\_position)

{

if (x < 9)

x++;

break;

}

else

{

if (x < 10)

x++;

break;

}

}

else if (number\_ship >= 7 && number\_ship < 9)

{

if (!number\_position)

{

if (x < 8)

x++;

break;

}

else

{

if (x < 10)

x++;

break;

}

}

else if (number\_ship == 9)

{

if (!number\_position)

{

if (x < 7)

x++;

break;

}

else

{

if (x < 10)

x++;

break;

}

}

}

case 's':

{

if (number\_ship < 4)

{

if (y < 10)

y++;

break;

}

else if (number\_ship >= 4 && number\_ship < 7)

{

if (!number\_position)

{

if (y < 10)

y++;

break;

}

else

{

if (y < 9)

y++;

break;

}

}

else if (number\_ship >= 7 && number\_ship < 9)

{

if (!number\_position)

{

if (y < 10)

y++;

break;

}

else

{

if (y < 8)

y++;

break;

}

}

else if (number\_ship == 9)

{

if (!number\_position)

{

if (y < 10)

y++;

break;

}

else

{

if (y < 7)

y++;

break;

}

}

}

case 13:

{

if (number\_ship < 4)

{

ship\_1 s1(x, y);

if ((\*this).CheckShip1(s1))

{

board[y - 1][x - 1] = false;

one\_deck.push\_back(s1);

one\_deck\_use.push\_back(s1);

number\_ship++;

}

break;

}

else if (number\_ship >= 4 && number\_ship < 7)

{

if (!number\_position)

{

ship\_2 s2(x, y, false);

if ((\*this).CheckShip2(s2))

{

board[y - 1][x - 1] = false;

board[y - 1][x] = false;

two\_deck.push\_back(s2);

two\_deck\_use.push\_back(s2);

number\_ship++;

}

break;

}

else

{

ship\_2 s2(x, y, true);

if ((\*this).CheckShip2(s2))

{

board[y - 1][x - 1] = false;

board[y][x - 1] = false;

two\_deck.push\_back(s2);

two\_deck\_use.push\_back(s2);

number\_ship++;

}

break;

}

}

else if (number\_ship >= 7 && number\_ship < 9)

{

if (!number\_position)

{

ship\_3 s3(x, y, false);

if ((\*this).CheckShip3(s3))

{

board[y - 1][x - 1] = false;

board[y - 1][x] = false;

board[y - 1][x + 1] = false;

three\_deck.push\_back(s3);

three\_deck\_use.push\_back(s3);

number\_ship++;

}

break;

}

else

{

ship\_3 s3(x, y, true);

if ((\*this).CheckShip3(s3))

{

board[y - 1][x - 1] = false;

board[y][x - 1] = false;

board[y + 1][x - 1] = false;

three\_deck.push\_back(s3);

three\_deck\_use.push\_back(s3);

number\_ship++;

}

break;

}

}

else if (number\_ship == 9)

{

if (!number\_position)

{

ship\_4 s4(x, y, false);

if ((\*this).CheckShip4(s4))

{

board[y - 1][x - 1] = false;

board[y - 1][x] = false;

board[y - 1][x + 1] = false;

board[y - 1][x + 2] = false;

four\_deck = s4;

four\_deck\_use = s4;

number\_ship++;

}

break;

}

else

{

ship\_4 s4(x, y, true);

if ((\*this).CheckShip4(s4))

{

board[y - 1][x - 1] = false;

board[y][x - 1] = false;

board[y + 1][x - 1] = false;

board[y + 2][x - 1] = false;

four\_deck = s4;

four\_deck\_use = s4;

number\_ship++;

}

break;

}

}

}

case ' ':

{

if (number\_ship >= 4 && number\_ship < 7)

{

if (x != 10 && y != 10)

{

if (!number\_position)

number\_position = true;

else

number\_position = false;

}

break;

}

else if (number\_ship >= 7 && number\_ship < 9)

{

if (x < 9 && y < 9)

{

if (!number\_position)

number\_position = true;

else

number\_position = false;

}

break;

}

else if (number\_ship == 9)

{

if (x < 8 && y < 8)

{

if (!number\_position)

number\_position = true;

else

number\_position = false;

}

break;

}

}

}

}

void DrawBoard(int & x, int & y, vector<vector<int>> & pc\_attack, vector<vector<int>> & v\_attack) //нарисовать доску пользователя и вспомагательную карту врага

{

cout << " ";

for (int i = 0; i < 22; i++)

{

if (i < 10)

cout << i + 1 << " ";

else if (i == 10 || i == 11)

cout << " ";

else

cout << " " << i - 11;

}

cout << endl;

for (int j = 0; j < 10; j++)

{

cout << j + 1;

for (int k = 0; k < 48; k++)

{

if (j == 9)

{

if (k == 47)

break;

if (k < 19)

{

if (k % 2 == 0 && pc\_attack[j][k / 2] == 2)

cout << "@";

else if (k % 2 == 0 && board[j][k / 2] == false)

cout << "\*";

else if (k % 2 == 0 && pc\_attack[j][k / 2] == 1)

cout << "-";

else

putchar(' ');

}

else if (k >= 19 && k < 27)

{

if (k == 26)

cout << j + 1;

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 - 12 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && v\_attack[j][k / 2 - 13] == 1)

cout << "-";

else if (k % 2 == 1 && v\_attack[j][k / 2 - 13] == 2)

cout << "@";

else

putchar(' ');

}

}

else

{

if (k <= 19)

{

if (k % 2 == 1 && pc\_attack[j][k / 2] == 2)

cout << "@";

else if (k % 2 == 1 && board[j][k / 2] == false)

cout << "\*";

else if (k % 2 == 1 && pc\_attack[j][k / 2] == 1)

cout << "-";

else

putchar(' ');

}

else if (k > 19 && k < 28)

{

if (k == 27)

cout << j + 1;

else

putchar(' ');

}

else

{

if (k % 2 == 1 && k / 2 - 13 == x && j + 1 == y)

cout << "+";

else if (k % 2 == 1 && v\_attack[j][k / 2 - 14] == 1)

cout << "-";

else if (k % 2 == 1 && v\_attack[j][k / 2 - 14] == 2)

cout << "@";

else

putchar(' ');

}

}

}

cout << endl;

}

}

int AttackUser(int & x, int & y, vector<vector<int>> & pc\_attack, vector<vector<int>> & v\_attack) //атака по компьютеру

{

int number, m;

switch (\_getch())

{

case 'a':

{

if (x > 1)

x--;

return -1;

break;

}

case 'w':

{

if (y > 1)

y--;

return -1;

break;

}

case 'd':

{

if (x < 10)

x++;

return -1;

break;

}

case 's':

{

if (y < 10)

y++;

return -1;

break;

}

case 13:

{

if (v\_attack[y - 1][x - 1] == 1 || v\_attack[y - 1][x - 1] == 2)

return -1;

else

{

number = GetAttackOnPc(x, y, v\_attack);

if (number == 0)

{

v\_attack[y - 1][x - 1] = 1;

return 0;

break;

}

else if (number == 1)

{

return 1;

break;

}

else

{

m = InfShipFromPc(x, y);

if (m == 1)

number\_1\_pc--;

else if (m == 2)

number\_2\_pc--;

else if (m == 3)

number\_3\_pc--;

else

number\_4\_pc--;

return 2;

break;

}

}

}

default:

{

return -1;

break;

}

}

}

bool GameOverUse()

{

if (number\_1\_use == 0 && number\_2\_use == 0 && number\_3\_use == 0 && number\_4\_use == 0)

return false;

else

return true;

}

};

class PlayerComputer : public PlayerSeaBattle

{

public:

static int temp;

static int control;

static int deck;

static int qwerty;

bool InitializeShipComputer() //инициализировать корабли у компьютера

{

one\_deck.clear();

two\_deck.clear();

three\_deck.clear();

one\_deck\_pc.clear();

two\_deck\_pc.clear();

three\_deck\_pc.clear();

for (int i = 0; i < 10; i++)

for (int j = 0; j < 10; j++)

board[i][j] = true;

srand(int(time(0)));

int random\_x = 0;

int random\_y = 0;

int random\_pos;

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

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 10 + 1;

if (i == 0)

{

ship\_1 s(random\_x, random\_y);

one\_deck.push\_back(s);

one\_deck\_pc.push\_back(s);

board[random\_y - 1][random\_x - 1] = false;

}

else

{

ship\_1 s(random\_x, random\_y);

if (!CheckShip1(s))

{

while (!CheckShip1(s))

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 10 + 1;

s.change(random\_x, random\_y);

}

}

one\_deck.push\_back(s);

one\_deck\_pc.push\_back(s);

board[random\_y - 1][random\_x - 1] = false;

}

}

for (int i = 0; i < 3; i++)

{

random\_pos = rand() % 2;

if (random\_pos == 0)

{

random\_x = rand() % 9 + 1;

random\_y = rand() % 10 + 1;

ship\_2 s(random\_x, random\_y, false);

if (!CheckShip2(s))

{

while (!CheckShip2(s))

{

random\_x = rand() % 9 + 1;

random\_y = rand() % 10 + 1;

s.change(random\_x, random\_y);

}

}

two\_deck.push\_back(s);

two\_deck\_pc.push\_back(s);

board[random\_y - 1][random\_x - 1] = false;

board[random\_y - 1][random\_x] = false;

}

else

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 9 + 1;

ship\_2 s(random\_x, random\_y, true);

if (!CheckShip2(s))

{

while (!CheckShip2(s))

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 9 + 1;

s.change(random\_x, random\_y);

}

}

two\_deck.push\_back(s);

two\_deck\_pc.push\_back(s);

board[random\_y - 1][random\_x - 1] = false;

board[random\_y][random\_x - 1] = false;

}

}

for (int i = 0; i < 2; i++)

{

random\_pos = rand() % 2;

if (random\_pos == 0)

{

random\_x = rand() % 8 + 1;

random\_y = rand() % 10 + 1;

ship\_3 s(random\_x, random\_y, false);

if (!CheckShip3(s))

{

while (!CheckShip3(s))

{

random\_x = rand() % 8 + 1;

random\_y = rand() % 10 + 1;

s.change(random\_x, random\_y);

temp++;

if (temp == 10)

return false;

}

}

three\_deck.push\_back(s);

three\_deck\_pc.push\_back(s);

board[random\_y - 1][random\_x - 1] = false;

board[random\_y - 1][random\_x] = false;

board[random\_y - 1][random\_x + 1] = false;

}

else

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 8 + 1;

ship\_3 s(random\_x, random\_y, true);

if (!CheckShip3(s))

{

while (!CheckShip3(s))

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 8 + 1;

s.change(random\_x, random\_y);

temp++;

if (temp == 10)

return false;

}

}

three\_deck.push\_back(s);

three\_deck\_pc.push\_back(s);

board[random\_y - 1][random\_x - 1] = false;

board[random\_y][random\_x - 1] = false;

board[random\_y + 1][random\_x - 1] = false;

}

}

random\_pos = rand() % 2;

if (random\_pos == 0)

{

random\_x = rand() % 7 + 1;

random\_y = rand() % 10 + 1;

ship\_4 s(random\_x, random\_y, false);

if (!CheckShip4(s))

{

while (!CheckShip4(s))

{

random\_x = rand() % 7 + 1;

random\_y = rand() % 10 + 1;

s.change(random\_x, random\_y);

temp++;

if (temp == 10)

return false;

}

}

four\_deck = s;

four\_deck\_pc = s;

board[random\_y - 1][random\_x - 1] = false;

board[random\_y - 1][random\_x] = false;

board[random\_y - 1][random\_x + 1] = false;

board[random\_y - 1][random\_x + 2] = false;

}

else

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 7 + 1;

ship\_4 s(random\_x, random\_y, true);

if (!CheckShip4(s))

{

while (!CheckShip4(s))

{

random\_x = rand() % 10 + 1;

random\_y = rand() % 7 + 1;

s.change(random\_x, random\_y);

temp++;

if (temp == 10)

return false;

}

}

four\_deck = s;

four\_deck\_pc = s;

board[random\_y - 1][random\_x - 1] = false;

board[random\_y][random\_x - 1] = false;

board[random\_y + 1][random\_x - 1] = false;

board[random\_y + 2][random\_x - 1] = false;

}

return true;

}

vector<int> ReturnCoordinate(int x, int y) //вернуть координату

{

vector<int> v;

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

if (one\_deck\_use[i].x1\_ship1 == x && one\_deck\_use[i].y1\_ship1 == y)

{

v.push\_back(x);

v.push\_back(y);

return v;

}

for (int j = 0; j < 3; j++)

if ((two\_deck\_use[j].x1\_ship2 == x && two\_deck\_use[j].y1\_ship2 == y) || (two\_deck\_use[j].x2\_ship2 == x && two\_deck\_use[j].y2\_ship2 == y))

{

v.push\_back(x);

v.push\_back(y);

return v;

}

for (int j = 0; j < 2; j++)

if ((three\_deck\_use[j].x1\_ship3 == x && three\_deck\_use[j].y1\_ship3 == y) || (three\_deck\_use[j].x2\_ship3 == x && three\_deck\_use[j].y2\_ship3 == y) || (three\_deck\_use[j].x3\_ship3 == x && three\_deck\_use[j].y3\_ship3 == y))

{

v.push\_back(x);

v.push\_back(y);

return v;

}

if (four\_deck\_use.x1\_ship4 == x && four\_deck\_use.y1\_ship4 == y)

{

v.push\_back(x);

v.push\_back(y);

return v;

}

if (four\_deck\_use.x2\_ship4 == x && four\_deck\_use.y2\_ship4 == y)

{

v.push\_back(x);

v.push\_back(y);

return v;

}

if (four\_deck\_use.x3\_ship4 == x && four\_deck\_use.y3\_ship4 == y)

{

v.push\_back(x);

v.push\_back(y);

return v;

}

if (four\_deck\_use.x4\_ship4 == x && four\_deck\_use.y4\_ship4 == y)

{

v.push\_back(x);

v.push\_back(y);

return v;

}

return v;

}

ship\_1 & ReturnShipFromUs\_1(int x, int y) //вернуть однопалубный корабль у пользователя

{

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

if (one\_deck\_use[i].x1\_ship1 == x && one\_deck\_use[i].y1\_ship1 == y)

return one\_deck\_use[i];

return one\_deck\_use[0];

}

ship\_2 & ReturnShipFromUs\_2(int x, int y)//вернуть двухпалубный корабль у пользователя

{

for (int j = 0; j < 3; j++)

if ((two\_deck\_use[j].x1\_ship2 == x && two\_deck\_use[j].y1\_ship2 == y) || (two\_deck\_use[j].x2\_ship2 == x && two\_deck\_use[j].y2\_ship2 == y))

return two\_deck\_use[j];

return two\_deck\_use[0];

}

ship\_3 & ReturnShipFromUs\_3(int x, int y)//вернуть трехпалубный корабль у пользователя

{

for (int j = 0; j < 2; j++)

if ((three\_deck\_use[j].x1\_ship3 == x && three\_deck\_use[j].y1\_ship3 == y) || (three\_deck\_use[j].x2\_ship3 == x && three\_deck\_use[j].y2\_ship3 == y) || (three\_deck\_use[j].x3\_ship3 == x && three\_deck\_use[j].y3\_ship3 == y))

return three\_deck\_use[j];

return three\_deck\_use[0];

}

int InfShipFromUs(int x, int y) // узнать, в какой корабль у пользователя попала пуля.

{

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

if (one\_deck\_use[i].x1\_ship1 == x && one\_deck\_use[i].y1\_ship1 == y)

return 1;

for (int j = 0; j < 3; j++)

if ((two\_deck\_use[j].x1\_ship2 == x && two\_deck\_use[j].y1\_ship2 == y) || (two\_deck\_use[j].x2\_ship2 == x && two\_deck\_use[j].y2\_ship2 == y))

return 2;

for (int j = 0; j < 2; j++)

if ((three\_deck\_use[j].x1\_ship3 == x && three\_deck\_use[j].y1\_ship3 == y) || (three\_deck\_use[j].x2\_ship3 == x && three\_deck\_use[j].y2\_ship3 == y) || (three\_deck\_use[j].x3\_ship3 == x && three\_deck\_use[j].y3\_ship3 == y))

return 3;

return 4;

}

int GetAttackOnUs(int x, int y, vector<vector<int>> & v\_enemy) //атака по пользователю

{

if (board\_enemy[y - 1][x - 1] == true)

{

v\_enemy[y - 1][x - 1] = 1;

return 0;

}

else

{

if (InfShipFromUs(x, y) == 1)

{

ReturnShipFromUs\_1(x, y).transformation1(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else if (InfShipFromUs(x, y) == 2)

{

ReturnShipFromUs\_2(x, y).testen\_two--;

if (ReturnShipFromUs\_2(x, y).testen\_two == 0)

{

ReturnShipFromUs\_2(x, y).transformation2(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else

{

v\_enemy[y - 1][x - 1] = 2;

return 1;

}

}

else if (InfShipFromUs(x, y) == 3)

{

ReturnShipFromUs\_3(x, y).testen\_three--;

if (ReturnShipFromUs\_3(x, y).testen\_three == 0)

{

ReturnShipFromUs\_3(x, y).transformation3(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else

{

v\_enemy[y - 1][x - 1] = 2;

return 1;

}

}

else if (InfShipFromUs(x, y) == 4)

{

four\_deck\_use.testen\_four--;

if (four\_deck\_use.testen\_four == 0)

{

four\_deck\_use.transformation4(v\_enemy);

v\_enemy[y - 1][x - 1] = 2;

return 2;

}

else

{

v\_enemy[y - 1][x - 1] = 2;

return 1;

}

}

}

return -1;

}

bool Shoot(int x, int y, vector<vector<int>> & v\_enemy) // проверка координаты

{

if (v\_enemy[y - 1][x - 1] == 0)

return true;

else

return false;

}

int AttackComputer(vector<int> & x\_vec, vector<int> & y\_vec, vector<vector<int>> & v\_enemy) //атака по пользователю

{

srand(int(time(0)));

int x, y, r, r\_copy, w, k;

if (x\_vec.size() < 1)

{

x = rand() % 10 + 1;

y = rand() % 10 + 1;

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 1;

return 1;

}

else

{

number\_1\_use--;

return 2;

}

}

else

{

if (control == 0)

{

x = rand() % 10 + 1;

y = rand() % 10 + 1;

if (!Shoot(x, y, v\_enemy))

{

while (!Shoot(x, y, v\_enemy))

{

x = rand() % 10 + 1;

y = rand() % 10 + 1;

}

}

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 1;

return 1;

}

else

{

number\_1\_use--;

return 2;

}

}

else if (control == 1)

{

r = rand() % 4 + 1;

while (r == 1 || r == 2 || r == 3 || r == 4)

{

if (r == 1)

{

x = x\_vec.back() - 1;

y = y\_vec.back();

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 2;

return 1;

}

else

{

number\_2\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else if (r == 2)

{

x = x\_vec.back() + 1;

y = y\_vec.back();

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 2;

return 1;

}

else

{

number\_2\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else if (r == 3)

{

x = x\_vec.back();

y = y\_vec.back() - 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 2;

return 1;

}

else

{

number\_2\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else

{

x = x\_vec.back();

y = y\_vec.back() + 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 2;

return 1;

}

else

{

number\_2\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

}

}

else if (control == 2)

{

r = rand() % 2 + 1;

while (r == 1 || r == 2)

{

if (deck > 5)

{

r = rand() % 4 + 1;

while(r==1 || r ==2 || r==3 || r==4)

{

if (r == 1)

{

k = InfShipFromUs(x\_vec.back(), y\_vec.back());

if (k == 3)

{

x = ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).x1\_ship3;

y = ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).y1\_ship3;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_3\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else

{

x = four\_deck\_use.x1\_ship4;

y = four\_deck\_use.y1\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

}

else if (r == 2)

{

k = InfShipFromUs(x\_vec.back(), y\_vec.back());

if (k == 3)

{

x = ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).x2\_ship3;

y = ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).y2\_ship3;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_3\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else

{

x = four\_deck\_use.x2\_ship4;

y = four\_deck\_use.y2\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

}

else if (r == 3)

{

k = InfShipFromUs(x\_vec.back(), y\_vec.back());

if (k == 3)

{

x = ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).x3\_ship3;

y = ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).y3\_ship3;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_3\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else

{

x = four\_deck\_use.x3\_ship4;

y = four\_deck\_use.y3\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

}

else

{

k = InfShipFromUs(x\_vec.back(), y\_vec.back());

if (k == 3)

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

else

{

x = four\_deck\_use.x4\_ship4;

y = four\_deck\_use.y4\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

}

}

}

else if (r == 1)

{

k = InfShipFromUs(x\_vec.back(), y\_vec.back());

if (k == 3)

{

if (ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).position3)

{

x = ReturnCoordinate(x\_vec.back(),y\_vec.back())[0];

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1] - 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 2;

return 1;

}

else

{

number\_3\_use--;

control = 0;

return 2;

}

}

else

{

deck++;

r = 2;

}

}

else

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0] - 1;

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1];

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 2;

return 1;

}

else

{

number\_3\_use--;

control = 0;

return 2;

}

}

else

{

deck++;

r = 2;

}

}

}

else

{

if (four\_deck\_use.position4)

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0];

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1] - 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

r = 2;

}

else

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0] - 1;

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1];

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

r = 2;

}

}

}

else

{

k = InfShipFromUs(x\_vec.back(), y\_vec.back());

if (k == 3)

{

if (ReturnShipFromUs\_3(x\_vec.back(), y\_vec.back()).position3)

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0];

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1] + 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_3\_use--;

control = 0;

return 2;

}

}

else

{

deck++;

r = 1;

}

}

else

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0] + 1;

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1];

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_3\_use--;

control = 0;

return 2;

}

}

else

{

deck++;

r = 1;

}

}

}

else

{

if (four\_deck\_use.position4)

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0];

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1] + 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

{

deck++;

r = 1;

}

}

else

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0] + 1;

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1];

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

control = 0;

return 2;

}

}

else

{

deck++;

r = 1;

}

}

}

}

}

}

else // control == 3 четырехпалубный

{

r = rand() % 2 + 1;

while (r == 1 || r == 2)

{

if (qwerty > 5)

{

r = rand() % 4 + 1;

while (r == 1 || r == 2 || r == 3 || r == 4)

{

if (r == 1)

{

x = four\_deck\_use.x1\_ship4;

y = four\_deck\_use.y1\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else if (r == 2)

{

x = four\_deck\_use.x2\_ship4;

y = four\_deck\_use.y2\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else if (r == 3)

{

x = four\_deck\_use.x3\_ship4;

y = four\_deck\_use.y3\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

else

{

x = four\_deck\_use.x4\_ship4;

y = four\_deck\_use.y4\_ship4;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 3;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

r\_copy = r;

r = rand() % 4 + 1;

if (r\_copy == r)

{

while (r\_copy == r)

{

r = rand() % 4 + 1;

}

}

}

}

}

}

else if (r == 1)

{

if (four\_deck\_use.position4)

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0];

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1] - 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 0;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

qwerty++;

r = 2;

}

}

else

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0] - 1;

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1];

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 0;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

qwerty++;

r = 2;

}

}

}

else

{

if (four\_deck\_use.position4)

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0];

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1] + 1;

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 0;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

qwerty++;

r = 1;

}

}

else

{

x = ReturnCoordinate(x\_vec.back(), y\_vec.back())[0] + 1;

y = ReturnCoordinate(x\_vec.back(), y\_vec.back())[1];

if (correctly(x, y) && Shoot(x, y, v\_enemy))

{

w = GetAttackOnUs(x, y, v\_enemy);

if (w == 0)

return 0;

else if (w == 1)

{

x\_vec.push\_back(x);

y\_vec.push\_back(y);

control = 0;

return 1;

}

else

{

number\_4\_use--;

control = 0;

return 2;

}

}

else

{

qwerty++;

r = 1;

}

}

}

}

}

}

return 0;

}

bool GameOverPc() //проверка на конец игры

{

if (number\_1\_pc == 0 && number\_2\_pc == 0 && number\_3\_pc == 0 && number\_4\_pc == 0)

return false;

else

return true;

}

};

void cout\_start() // функция вывода 1

{

cout << "Инициализируйте свои корабли!\n" << endl;

cout << "Управление курсором: a - \"влево\", w - \"вверх\", d - \"вправо\", s - \"вниз\"" << endl;

cout << "Чтобы повернуть корабль, используйте Space." << endl;

cout << "Чтобы установить корабль на выбранную позицию, нажмите Enter.\n" << endl;

}

void cout\_end(int stat, string & s1) //функция вывода 2

{

cout << "ИГРА НАЧАЛАСЬ!\n" << endl;

cout << "Обозначения: \"-\" - промах, \"@\" - попадание." << endl;

cout << "Через \"\*\" обозначены Ваши корабли, в которые еще не попала пуля." << endl;

cout << "Управление курсором: a - \"влево\", w - \"вверх\", d - \"вправо\", s - \"вниз\"" << endl;

cout << "Чтобы атаковать выбранную координату, нажмите Enter." << endl;

cout << " Оставшиеся корабли:" << endl;

cout << "Пользователь: Компьютер:" << endl;

cout << "Однопалубные:" << PlayerSeaBattle::number\_1\_use << " " << "Однопалубные:" << PlayerSeaBattle::number\_1\_pc << endl;

cout << "Двухпалубные:" << PlayerSeaBattle::number\_2\_use << " " << "Двухпалубные:" << PlayerSeaBattle::number\_2\_pc << endl;

cout << "Трехпалубные:" << PlayerSeaBattle::number\_3\_use << " " << "Трехпалубные:" << PlayerSeaBattle::number\_3\_pc << endl;

cout << "Четырехпалубные:" << PlayerSeaBattle::number\_4\_use << " " << "Четырехпалубные:" << PlayerSeaBattle::number\_4\_pc << endl;

if (stat == 1)

cout << "Состояние : ВАШ ХОД! - " << s1 << endl;

else if (stat == 2)

cout << "Состояние : ХОД КОМПЬЮТЕРА! - " << s1 << endl;

else

cout << "Состояние : ИГРА ЗАВЕРШИЛАСЬ!" << endl;

}

// ИНИЦИАЛИЗАЦИЯ СТАТИЧЕСКИХ ПЕРЕМЕННЫХ

int PlayerSeaBattle::number\_1\_use = 4;

int PlayerSeaBattle::number\_2\_use = 3;

int PlayerSeaBattle::number\_3\_use = 2;

int PlayerSeaBattle::number\_4\_use = 1;

int PlayerSeaBattle::number\_1\_pc = 4;

int PlayerSeaBattle::number\_2\_pc = 3;

int PlayerSeaBattle::number\_3\_pc = 2;

int PlayerSeaBattle::number\_4\_pc = 1;

int PlayerUser::number\_ship = 0;

bool PlayerUser::number\_position = true;

int PlayerComputer::temp = 0;

int PlayerComputer::control = 0;

int PlayerComputer::deck = 0;

int PlayerComputer::qwerty = 0;

vector<ship\_1> PlayerSeaBattle:: one\_deck\_use;

vector<ship\_2> PlayerSeaBattle:: two\_deck\_use;

vector<ship\_3> PlayerSeaBattle::three\_deck\_use;

ship\_4 PlayerSeaBattle:: four\_deck\_use;

vector<ship\_1> PlayerSeaBattle::one\_deck\_pc;

vector<ship\_2> PlayerSeaBattle:: two\_deck\_pc;

vector<ship\_3> PlayerSeaBattle::three\_deck\_pc;

ship\_4 PlayerSeaBattle::four\_deck\_pc;

int main()

{

setlocale(LC\_ALL, "Russian");

PlayerComputer pc;

PlayerUser use;

int a;

int x = 5, y = 5;

int b = -1;

int at;

string status = " ";

bool t = false;

vector<vector<int>> save\_attack\_use(10) , save\_attack\_pc(10);

vector<int> x\_save\_pc, y\_save\_pc;

for (int i = 0; i < 10; i++)

for (int j = 0; j < 10; j++)

{

save\_attack\_use[i].push\_back(0);

save\_attack\_pc[i].push\_back(0);

}

cout << "Морской бой!\n" << endl;

cout << "1. Начать игру." << endl;

cout << "2. Выйти." << endl;

cout << ">>";

cin >> a;

system("cls");

if (a == 1)

{

while (PlayerUser::number\_ship != 10)

{

cout\_start();

use.InitializeShip(x, y);

if (PlayerUser::number\_ship <= 9)

setcur(0, 0);

}

cout << endl;

cout << "Корабли установлены. Подождите, противник устанавливает свои корабли..." << endl;

\_sleep(2000);

pc.GetBoard(use.ReturnBoard());

if (!pc.InitializeShipComputer())

while (!t)

{

PlayerComputer::temp = 0;

t = pc.InitializeShipComputer();

}

use.GetBoard(pc.ReturnBoard());

system("cls");

x = 5, y = 5;

while (use.GameOverUse() == true && pc.GameOverPc() == true)

{

while (b == -1)

{

cout\_end(1, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

if (pc.GameOverPc() && use.GameOverUse())

{

b = use.AttackUser(x, y, save\_attack\_pc, save\_attack\_use);

}

else

break;

setcur(0, 0);

}

if (b == 1 || b == 2)

{

while (b == 1 || b == 2)

{

if (b == 1)

status = "Попадание!";

else

status = "Корабль уничтожен!";

cout\_end(1, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

if (pc.GameOverPc() && use.GameOverUse())

{

b = use.AttackUser(x, y, save\_attack\_pc, save\_attack\_use);

}

else

break;

if (b == -1 && status == "Попадание!")

b = 1;

else if (b == -1 && status == "Корабль уничтожен!")

b = 2;

system("cls");

}

}

if (b == 0)

{

status = "Мимо!";

cout\_end(1, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

\_sleep(2000);

system("cls");

}

if (b != 0)

\_sleep(2000);

status = " ";

cout\_end(2, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

\_sleep(2000);

if (pc.GameOverPc() && use.GameOverUse())

{

at = pc.AttackComputer(x\_save\_pc, y\_save\_pc, save\_attack\_pc);

}

else

break;

setcur(0, 0);

if (at == 1 || at == 2)

{

while (at == 1 || at == 2)

{

if (at == 1)

status = "Попадание!";

else

status = "Корабль уничтожен!";

cout\_end(2, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

\_sleep(2000);

if (pc.GameOverPc() && use.GameOverUse())

{

at = pc.AttackComputer(x\_save\_pc, y\_save\_pc, save\_attack\_pc);

}

else

break;

system("cls");

}

}

if (at == 0)

{

status = "Мимо!";

cout\_end(2, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

\_sleep(2000);

system("cls");

}

b = -1;

status = " ";

}

system("cls");

status = " ";

if (!use.GameOverUse())

{

cout\_end(3, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

cout << endl;

cout << "Игра закончена! Вы програли." << endl;

}

else

{

cout\_end(3, status);

use.DrawBoard(x, y, save\_attack\_pc, save\_attack\_use);

cout << endl;

cout << "Игра закончена! Вы выиграли." << endl;

}

}

else

return 0;

}