**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Объекто-ориентированное программирование»**

Тема: Конструкторы и деструкторы

|  |  |  |
| --- | --- | --- |
| Студент гр. 2382 |  | Ваньков Я.С. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

## Цель работы

Изучить методы работы с классами. Написать простую программу в стиле ООП. Изучить работу конструктора и деструктора в С++. Изучить конструкторы перемещения и конструкторы копирования.

## **Задание**

а) Создать класс клетки игрового поля. Клетка игрового поля может быть проходимой или нет, тем самым определяя возможность игрока встать на эту клетку. Возможность задать проходимость клетки должна быть реализована через конструктор и через метод клетки. В будущем в клетке будет храниться указатель на интерфейс события.

б) Создать класс игрового поля. Игровое поле представляет собой прямоугольник из клеток (двумерный массив). В учебных целях, клетки хранятся как чистый массив на указателях (использовать контейнеры stl запрещено в этой лаб. работе). Размер поля передается в конструктор поля, в котором динамически выделяется память под массив клеток. Также должна быть возможность вызвать конструктор поля без аргументов. Так как происходит выделение память, то необходимо реализовать деструктор в котором будет происходить очистка память. Также для класса поля необходимо реализовать конструкторы копирования и перемещения, а также соответствующие им операторы присваивания.

Также добавить в игровое поле добавьте информацию о входе (где в начале появляется игрок) и выходе (куда игрок должен дойти)

в) В класс управления игрока добавить взаимодействия с полем. При перемещении игрока должна быть проверка на проходимость клетки, если клетка непроходима, то перемещение не должно производиться.

Примечания:

Так как в клетке будет храниться указатель, то при копировании и перемещении должен быть предусмотрен механизм копирования объекта хранящегося по указателю.

Убедитесь, что присутствует проверка контроля размера поля, чтобы его нельзя было сделать слишком маленьким или с отрицательными размерами.

В конструкторе перемещения и соответствующем ему операторе присваивания не должно происходить никакого копирования данных.

Через класс поля должен быть доступ к каждой индивидуальной клетке. Создавать метод, который возвращает указатель на весь массив или указатель на каждую клетку, плохая практика, так как появляется возможность очистки памяти.

## Выполнение работы

**Player — класс игрока**

Класс игрока. В публичных полях находятся ссылки на все характеристики игрока. Не имеет методов.

**Characterisric — класс характеристики**

Главный класс для характеристик: от него наследуются все характеристики. В конструктор класса передается минимальное, максимальное и установленное значение.

Определено три метода:

getValue — возвращает значение

setValue — меняет значение, учитывая ограничение класса.

Check — проеверяет измененное значение, согласно ограничениям класса

Оператор присваивания перегружен — при попытки присваивания выбрасывает ошибку

**Наследники класса характеристик:**

**Health — класс здоровья**

Добавляется приватное поле игрока — is\_dead, которое принимает только булевые типы данных.

Также добавляется метод на проверку наличия здоровья у игрока.

**Armor — класс брони**

**Count — класс количества**

**Money — класс денег**

**Power — класс силы**

**Score — класс очков**

**Item. - класс предмета**

В приватных полях лежит название предмета и его характеристики(они же и инициализируются в конструкторе).

Методы:

getName — вовзвращает имя

setName — изменяет имя

**Weapon — класс оружия**

В приватных полях класса лежит урон, наносимый оружием и название оружия.

Методы класса:

getDamage — вовзвращает урон

getName — вовзвращает имя

setDamage — изменяет урон

setName — изменяет имя

use — изменяет значение силы игрока на значение урона

**Coordinate** **— класс координаты**

В приватных полях имеет координаты по X и Y.

Дружественный класс для MoveManager и Square

Публичные методы:

getX — возвращает X

getY — возвращает Y

getPair — возвращает пару

**MoveManager — класс контроллера над игроком**

Класс, отвечающий за контроль движения игрока на поле.

Конструктор класса принимает игрока, над которым будут производится действия и поле, на котором будет находиться игрока.

В приватных полях находится ссылка на игрока, над которым воздействует менеджер и ссылка поле на котором, находится игрок. Также в приватном поле находится координата игрока.

Методы в зависимости от переданного значения изменяют координату игрока.

Реализован метод checkCoord, который возвращает булевое значение - запрета перемещения в клетку, где стоит стена — или выход за пределы поля.

**Field — класс поля**

Конструктор принимает высоту и ширину, а также координаты старта и конца. Каждый аргумент имеет дефолтное значение, поэтому можно создать поле, не указывая аргументов в конструкторе. Поле проверяется на допустимость размера. Выделяется память под поле.

Реализован конструктор копирования, перемещения и деструктор.

Публичные методы:

getSquare — перегруженный метод, который может принимать координаты, а также целочисленные значения x и y. Так реализуется приницы полиморфизма. Возвращает ссылка на клетку

GetHeight и getWidth — методы возвращающие числовое значение высоты и ширины.

GetStart и getEnd — методы, возвращающие ссылки на координаты старта и конца.

CheckSize — перегруженный метод, который может принимать координаты, а также целочисленные значения x и y. Так реализуется приницы полиморфизма. Возвращает булевое значение корректности размера поля

**Square**

Констуктор принимает проходимость клетки и ее координату.

В приватных полях класса находится координата и проходимость.

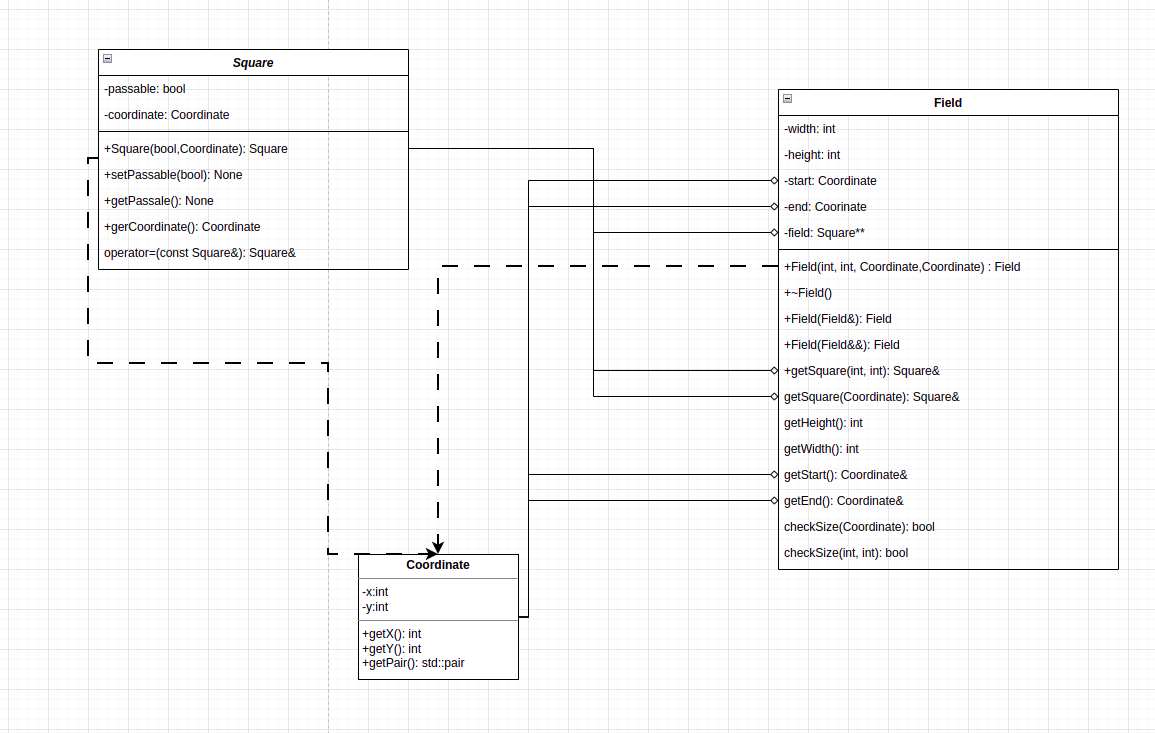
Реализованы следующие методы класса:

setPassable — задает проходимость указанной клетки

getPassable — возвращает проходимость указанной клетки

getCoordinate — возвращает ссылку на координату указанной клетки

оператор присваивания перегружен — присвоить клетку можно только в случае, если у нее задано дефолтное значение.



## Выводы

Были изучены методы работы с классами. Были написаны классы игрока, его характеристик и классы взаимодействия игрока с его координатами. Был написан класс поля и клетки. К классу поле реализованы конструкторы копирования и перемешение, а также деструктор.

# Приложение А Исходный код программы

# https://github.com/DoAlvaro/OOP\_2382