МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Объектно-ориентированное программирование»

Студент гр. 4384 Кочкин Д. М.

Преподаватель Жангиров Т. Р.

Санкт-Петербург

2025

Цель работы.

Изучение принципов объектно-ориентированного проектирования и реализация игровой логики на языке C++. В ходе работы создаются классы для представления сущностей игрового поля (игрока, врагов, зданий, клеток и самого поля) и реализуются их взаимодействия.

Задание.

Создать класс игрока, который должен хранить информацию об игроке (его жизни, урон, очки, и т.д. - студент сам определяет необходимые для работы характеристики). Объект класса игрока должен перемещаться по карте. Если у игрока кончаются жизни, то происходит конец игры.

Создать класс врага, который хранит параметры жизней и урона. Объектами класса врага управляет компьютер. При перемещении, если враг пытается перейти на клетку с игроком, то перемещение не происходит, и игроку наносится урон.

Создать класс квадратного/прямоугольного игрового поля, по которому перемещаются игрок и враги. Игровое поле не должно быть меньше 10 на 10 клеток, и не больше 25 на 25 клеток. Размеры поля задаются через конструктор. Рекомендуется для хранения информации об отдельных клетках поля создать отдельный класс.

Реализовать конструкторы перемещения и копирования для поля, а также соответствующие операторы присваивания с копированием и перемещением (должна происходить глубокая копия).

Реализовать непроходимые клетки на поле. При попытке врагов или игрока перейти на такую клетку, перемещение не происходит. Заполнение поля непроходимыми клетками происходит в момент создания поля.

Добавить возможность для игрока переключаться на ближний или дальний бой с изменением значения наносимого урона. Такое переключение требует один ход.

Добавить класс вражеского здания. Такое здание размещается на карте, и раз в несколько ходов создает нового врага возле себя. Количество ходов до создания нового врага задается в конструкторе.

Реализовать замедляющие клетки на поле. Если игрок переходит на такую клетку, то он он не может двигаться на следующий ход.

Описание архитектуры программы.

Программа построена по принципам объектно-ориентированного программирования с чётким разделением ответственности между классами. Центральным элементом является класс Field, отвечающий за состояние игрового поля, размещение и обновление сущностей.

Основные классы:

• Entity — базовый абстрактный класс, определяющий общие свойства всех сущностей (идентификатор, здоровье, урон).

• Player — класс игрока, наследник Entity. Реализует перемещение, режимы боя (ближний/дальний), взаимодействие с замедляющими клетками и врагами.

• Enemy — противник, движущийся к игроку. Если враг пытается шагнуть на клетку с игроком, он не перемещается, но наносит урон.

• Building — структура, периодически создающая врагов.

• Field — класс, содержащий сетку клеток, список сущностей и их координаты.

Реализует механику игры: обновление, движение, спавн врагов и проверку конца игры.

• Cell — элемент поля, хранящий тип содержимого (пустая, непроходимая, замедляющая и т.д.).

Связи классов: Player, Enemy и Building наследуют Entity. Field управляет их жизненным циклом. Класс Field хранит коллекции всех объектов и координирует взаимодействие между ними.

UML-диаграмма классов.

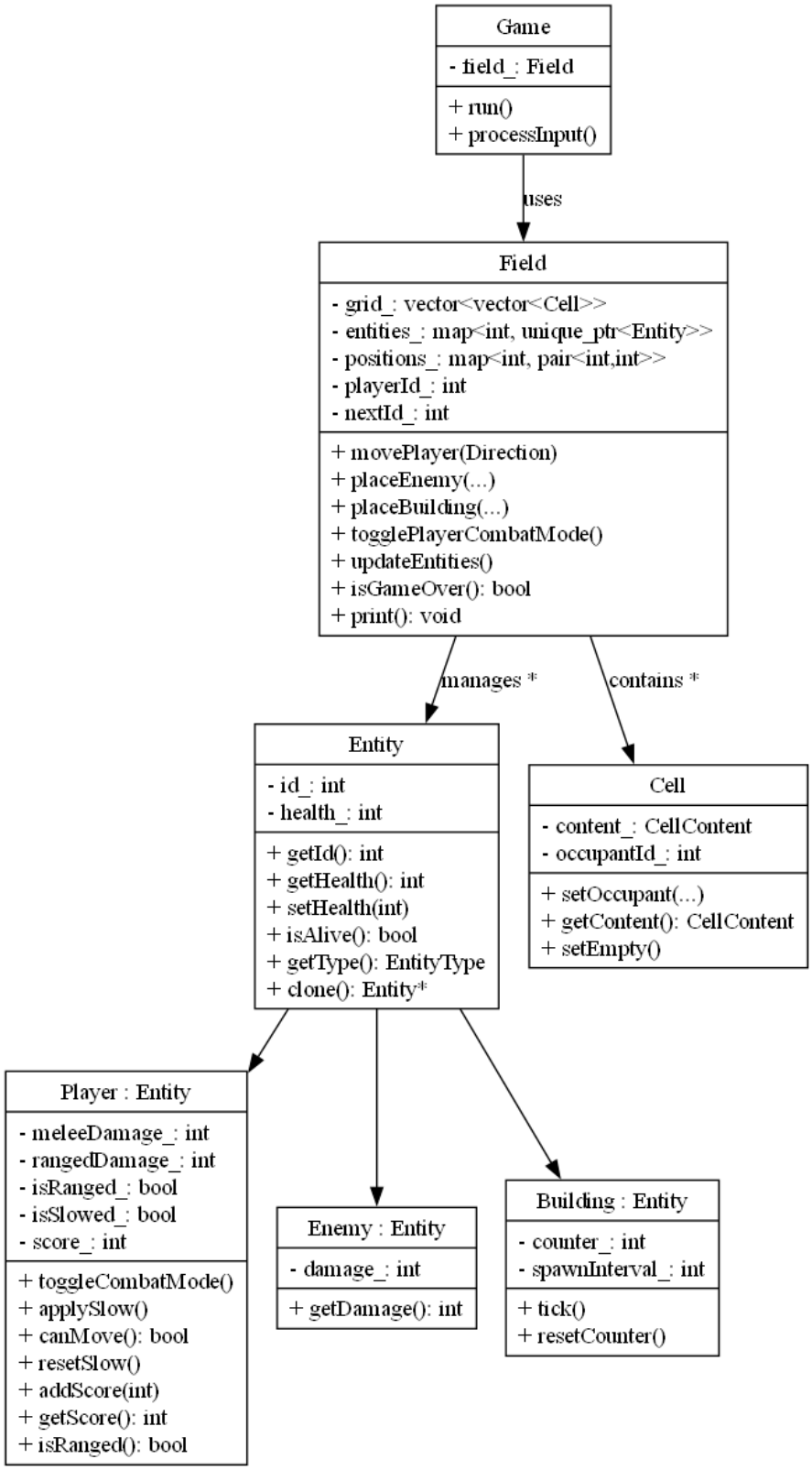

Рисунок 1 - UML-диаграмма

Проверка работы программы.

Была проведена проверка работы игры. Игрок может перемещаться по полю, сталкиваться с врагами и зданиями, а также переключаться между ближним и дальним боем. При переходе на клетку с врагом он атакует противника, но теряет возможность хода на следующий раунд. При переходе на замедляющую клетку — пропускает ход. Враги корректно двигаются к игроку и атакуют при попытке занять его клетку.

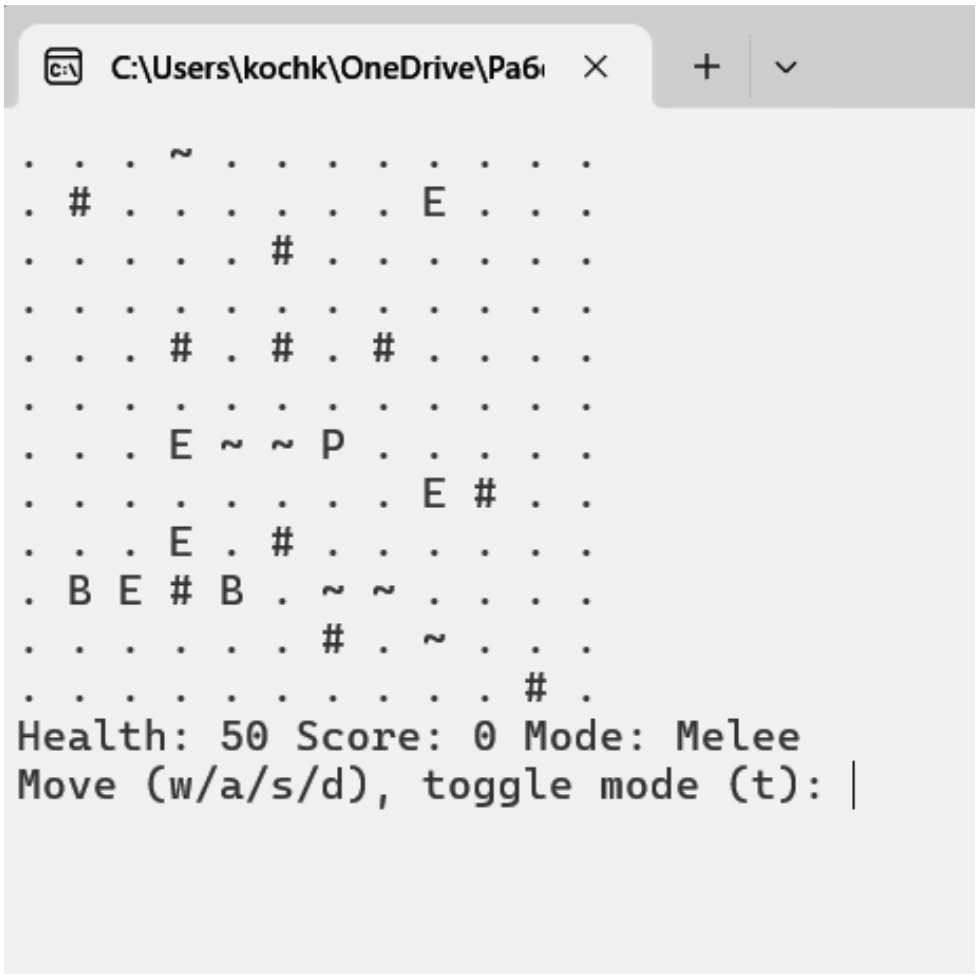

Рисунок 2 - Игровое поле

Выводы.

В результате выполнения лабораторной работы были изучены принципы ООП: наследование, инкапсуляция и полиморфизм. Разработана архитектура игры с несколькими типами сущностей и взаимодействиями между ними. Программа успешно проходит тестирование и демонстрирует корректное поведение объектов на игровом поле.