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

ОТЧЕТ

по лабораторной работе № 1

по дисциплине «Объектно ориентированное программирование»

Тема: Создание классов, конструкторов и методов класса

Студент гр. 0383 _____ Коротков А.В.

Преподаватели _____ Жангиров Т.Р.

Санкт-Петербург
2021

Цель работы.

Реализовать классы “клетка” и “поле”. Для класса клетки необходимо сделать интерфейс, а для поля конструкторы копирования и перемещения, а также соответствующие операторы.

Задание.

Игровое поле представляет из себя прямоугольную плоскость разбитую на клетки. На поле на клетках в дальнейшем будут располагаться игрок, враги, элементы взаимодействия. Клетка может быть проходимой или непроходимой, в случае непроходимой клетки, на ней ничего не может располагаться. На поле должны быть две особые клетки: вход и выход. В дальнейшем игрок будет появляться на клетке входа, а затем выполнив определенный набор задач дойти до выхода.

При реализации класса поля запрещено использовать контейнеры из stl

Требования:

- Реализовать класс поля, который хранит набор клеток в виде двумерного массива.
- Реализовать класс клетки, которая хранит информацию о ее состоянии, а также того, что на ней находится.
- Создать интерфейс элемента клетки.
- Обеспечить появление клеток входа и выхода на поле. Данные клетки не должны быть появляться рядом.
- Для класса поля реализовать конструкторы копирования и перемещения, а также соответствующие операторы.
- Гарантировать отсутствие утечки памяти.

Потенциальные паттерны проектирования, которые можно использовать:

Итератор (Iterator) - обход поля по клеткам и получение косвенного доступа к ним

Строитель (Builder) - предварительное конструирование поля с необходимым параметрами. Например, предварительно задать кол-во непроходимых клеток и алгоритм их расположения

Выполнение работы.

Был реализован класс Game, отвечающий за всю бизнес-логику игры и содержащий в себе *sf::RenderWindow* и *sf::Event* для работы с окном и управлением, а также поле класса Field и класс, отвечающий за его отрисовку - Drawer, созданный для разделения бизнес-логики и пользовательского интерфейса. Для запуска бизнес-логики используется функция *void run()*.

Для инициализации карты игрового поля в классе предусмотрена функция загрузки шаблонов поля из текстового файла. Из представленных в файле шаблонов в ходе инициализации класса Game случайным образом выбирается один из шаблонов.

Главный цикл игры внутри класса был разделён на функции *void updateEvents()*, предназначенной для обработки событий, таких как ввод с клавиатуры или нажатие мыши, *void updateLogic()*, содержащей любые обновления логики игрового процесса, а также *void render()*, в процессе которой производится работа с окном, отрисовка поля с помощью вышеуказанного класса *Drawer*.

Был реализован класс *Field*, отвечающий за бизнес-логику игрового поля. Данный класс содержит в себе двумерный массив экземпляров класса Tile, представляющих собой отдельные клетки поля. Также класс хранит информацию о размерах поля, реализует операторы и конструкторы копирования и присваивания и представляет возможность получить доступ к размерам поля и конкретной клетке по её координатам с помощью “get`теров”.

Каждая клетка представляет собой контейнер для хранения некоторой сущности (переменная entity), находящейся на клетке в данный момент, и предоставляет интерфейс для работы с текущим состоянием клетки. С помощью реализованных функций *bool isEmpty(), bool isPassable(), unsigned short getType()* и *const IEntity& getEntity()* возможно получить доступ к информации о том, пуста ли клетка, является ли она доступной для перемещения по ней каким-либо персонажем либо врагом, а также о том, какой клетка имеет тип. Всего предусмотрено 4 типа клеток - пустая клетка, стена, вход и выход. Непроходимым считается лишь тип клетки “стена”. С помощью последней из перечисленных функций класс позволяет обратиться по неизменяемой ссылке к объекту класса IEntity.

Класс IEntity представляет собой интерфейс объекта, способного находится на клетке. В дальнейшем от данного класса будут наследоваться классы персонажа, врагов, а также предметов.

Для работы с графикой при выполнении работы была выбрана библиотека SFML.

UML диаграмму см. в приложении А.

Выводы.

В ходе выполнения лабораторной работы была написана программа, реализующая классы клетки и поля, а также конструкторы копирования и присваивания и соответствующие операторы.

Приложение А

UML-диаграмма