**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. 0382 _____ Тихонов С.В.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

**Цель работы.**

Изучение основных принципов работы с классами, создания конструкторов и методов классов на языке C++. Создание классов игрового поля и клетки и их методов.

**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.**

Сначала был создан класс клетки Cell(Объявление в Cell.h, реализация функций в Cell.cpp). Названия полей и методов всех классов представлены на UML диаграмме (см. приложение А).

В модуле с классом Cell содержится перечисление Type, Element в виде enum классов. . В перечислении существуют типы:

Types:

WALL

NORMAL

ENTRY

EXIT

Element:

WALL

NO_WALL

PLAYER

ENEMY

ITEM

ENEMY_AND_ITEM

Так же содерджится обычный класс Object* object это заготовка для следующих лабораторных работ, данный класс будет интерфейсом для того, что находится в этой клетке.

Описание принципов работы методов:

1. Конструктор инициализирует поля объекта, теми значениями Type и Elem, которые в него передали.
2. Метод setType необходим для изменения типа клетки. Принимает объект перечисления типа Types.

3. Метод setElem необходим для изменения елемента клекти, принимает объект перечисления типа Elem
4. Метод getType возвращает значения _type клетки, это нужна для проверки можно ли пройти через клетку и является она входом или выходом.
5. Метод getElem возвращает значение _elem клетки, чтобы узнать, что в ней находится.

Далее был реализован класс поля Field. В нём реализован двумерный объектов класса Cell – Cell** cell.

Описание принципов работы методов:

1. Конструктор. Конструктор в двойном цикле выделяет память при помощи new.
2. Конструктор копирования. Здесь в цикле for происходит копирование всех строк массива cell одного объекта в массив cell другого объекта.
3. Конструктор перемещения. Перемещает данные всех полей cell удаля старые ячейки.
4. Деструктор. Сначала освобождает память каждой строки, потом всего массива.
5. Метод getCell() возвращает двумерный массив клеток.
6. Метод printField() выводит в консоль поле в виде где, каждая ячейка состоит из двух цифр первая отвечает за элемент в данной клетке, вторая за её тип. Они разделены запятой.

Для создания объекта Field реализован класс fieldbildertree – реализация интерфейса FieldBuilder.

Описание принципов работы методов:

1. Конструктор. Инициализирует поле field объекта, выделяет память под него. В этом поле хранится указатель на создаваемый объект класса Field.
2. Деструктор. Освобождает выделенную память.
3. Метод Reset. Заменяет указатель field на новый. Важно не вызывать этот метод пока адрес создаваемого объекта не присвоен никакой внешней переменной, чтобы не потерять выделенную память.

4. Методы fieldbildertree, -GeneratNO_wallCells, GeneratWallCells, GeneratWallDungeonsCells -GeneratEntryCell GeneratExitCell. Используются для инициализации обычных клеток, входа, выхода и стен соответственно. Вход создаётся в левой верхней клетке, выход в правой нижней. Клетка стены генерируется псевдослучайным образом по одной на каждый квадрат из 4-х клеток, а так же вокруг карты. Остальные клетки – обычные.
5. Метод get_field. Возвращает адрес созданного объекта, передаёт управление над ним и вызывает метод Reset для того чтобы случайно не испортить уже созданный объект в процессе создания нового.

Для того, чтобы была возможность использования нескольких билдеров для разных типов полей (например, большего размера, или с другим алгоритмом расположения стен) реализован класс Fielddirector. Этот класс хранит указатель builder на объект класса FieldBuilder (интерфейс, реализациями которого являются конкретные билдеры). Имеется setbuilder для изменения используемого билдера. Реализован метод bilder_Fieldbildertree, который вызывает Build методы билдера. В дальнейшем могут быть созданы методы для создания других типов полей.

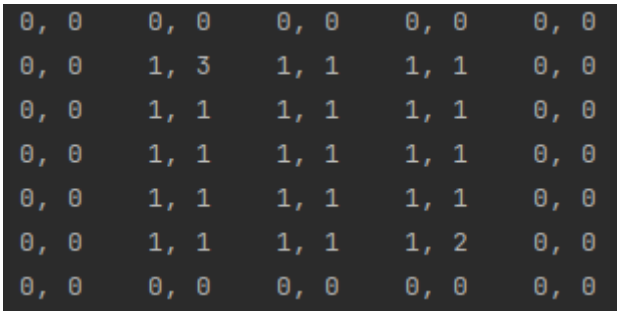
Был создан класс Start для того, чтобы создавать поле, а так же выводить его на экран.

**Тестирование.**

WIDTH=10 HIGHT=10

```
0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0
0, 0    1, 3    1, 1    1, 1    1, 1    1, 1    0, 0    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    1, 1    1, 1    0, 0    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    1, 1    1, 1    0, 0    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    1, 1    1, 1    1, 1    1, 1    1, 1    0, 0
0, 0    0, 0    0, 0    1, 1    0, 0    0, 0    0, 0    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    1, 1    1, 1    0, 0    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    1, 1    1, 1    0, 0    1, 1    1, 1    0, 0
0, 0    1, 1    1, 1    1, 1    1, 1    1, 1    0, 0    1, 1    1, 2    0, 0
0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0    0, 0
```

WIDTH =5 HIGHT=7




Результат тестирования

По итогам тестирования можно заметить, что конструирование и отображение поля и клеток работает правильно.

**Выводы.**

В ходе работы были изучены основные принципы работы с классами, создания конструкторов и методов классов на языке C++. Были созданы классы игрового поля и клетки и их методы. Также было проведено тестирование программы и применен паттерн программирования Builder.

# ПРИЛОЖЕНИЕ А

# UML ДИАГРАММА