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 _____ Кондратов Ю.А.

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. Названия полей и методов всех классов представлены на UML диаграмме (см. приложение А).

В модуле с классом Cell содержится перечисление Types, необходимое для наглядного представления типа клетки. В перечислении существуют типы: WALL = 0, FOUNDATION = 1, EXIT = 2, ENTRY = 3.

Описание принципов работы методов:

1. Конструктор инициализирует поля объекта, по умолчанию поле type инициализируется как FOUNDATION, а поле object как nullptr.
2. Метод set_type необходим для изменения типа клетки. Принимает объект перечисления Types.
3. Перегрузка оператора присваивания. В поле object копируется значение из поля object копируемого объекта. Полю type присваивается значения поля type копируемого объекта.
4. Метод is_passable. Возвращает значение поля type преобразованное к типу bool. Клетка непроходима в случае, если является стеной. Тогда поле type имеет значение 0 и метод возвращает false.

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

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 grid одного объекта в массив grid другого объекта.
3. Конструктор перемещения. Реализован при помощи функции std::swap.

4. Операторы присваивания с копированием и присваивания с перемещения организованы аналогично соответствующим конструкторам.
5. Деструктор. Сначала освобождает память каждой строки, потом всего массива.

Для создания объекта Field реализован класс PlainFieldBuilder – 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 BuildFoundation, -Entry, -Exit, -Walls. 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. 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 set_builder для изменения используемого билдера. Реализован метод buil_plain_field, который вызывает Build методы билдера. В дальнейшем могут быть созданы методы для создания других типов полей.

Класс CellObject является интерфейсом объекта клетки. Пока в нём реализованы только методы generate и remove для появления и удаления объекта из клетки соответственно. В дальнейшем при разработке логики игры будут реализованы и другие методы.

Тестирование.

Для тестирования программы реализован специальный код (см. рисунок 1).

```
auto builder = new PlainFieldBuilder();
auto f_director = FieldDirector(builder);
f_director.build_plain_field();
Field *field1 = builder->get_field();
f_director.build_plain_field();
Field *field2 = builder->get_field();
field1->draw_field(&window);
window.display();
sleep(3);
field2->draw_field(&window);
window.display();
sleep(3);
*field2 = *field1;
delete field1;
field2->draw_field(&window);
window.display();
sleep(5);
f_director.build_plain_field();
field1 = builder->get_field();
field1->draw_field(&window);
window.display();
sleep(3);
*field1 = std::move(*field2);
field1->draw_field(&window);
window.display();
sleep(3);
field2->draw_field(&window);
window.display();
sleep(5);
```

Рисунок 1 – Тестирующий код

В результате тестирования были выведены следующие версии полей (см. рисунок 2).

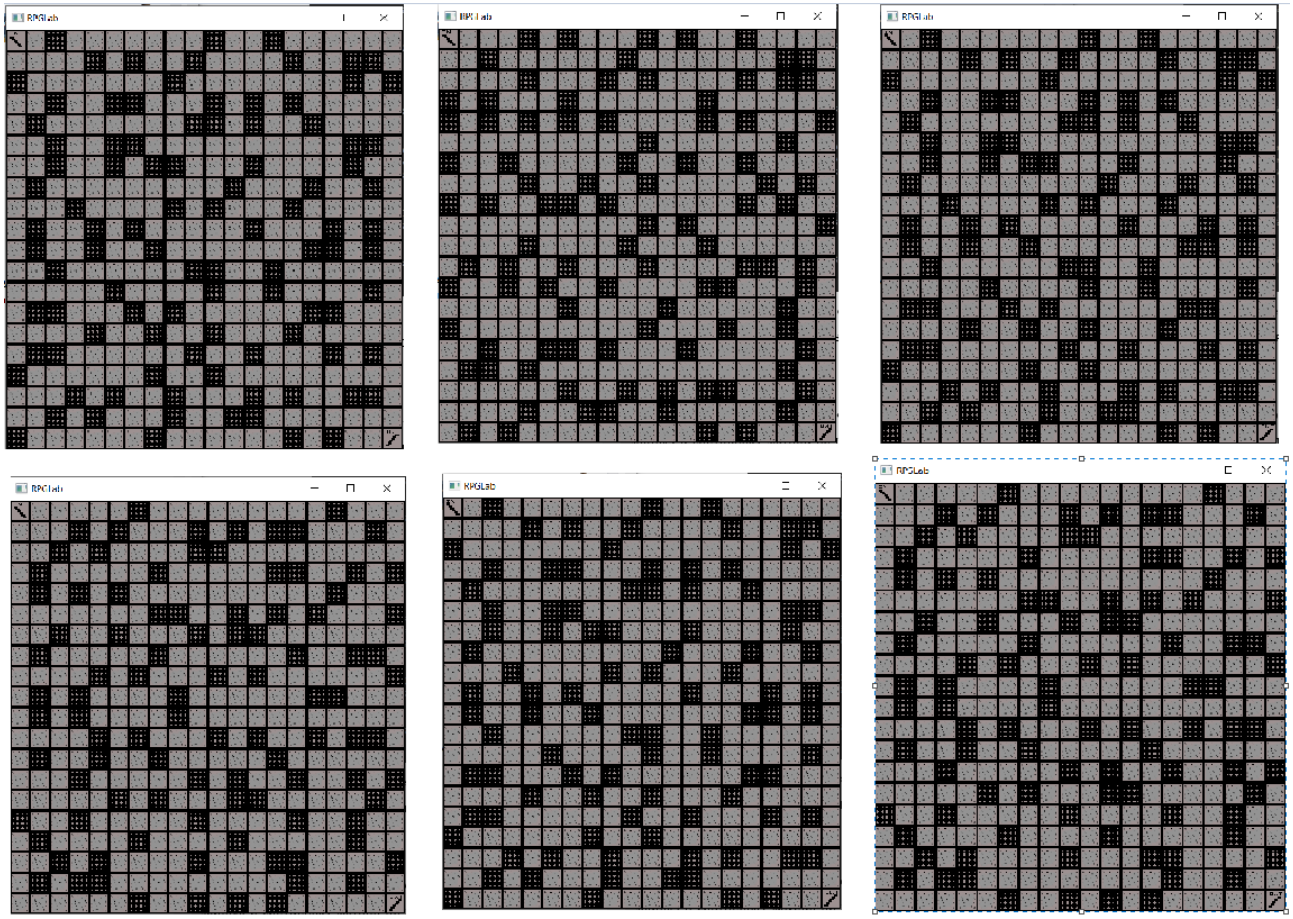

Рисунок 2 – Результат тестирования

По итогам тестирования можно заметить, что конструирование и отображение поля и клеток работает правильно, также правильно работают операторы перемещения и копирования.

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