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

ОТЧЕТ

по лабораторной работу №1

по дисциплине «Объектно-ориентированное программирование»

Тема: Создание классов, конструкторов и методов классов

Студентка гр. 0382 _____ Деткова А.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Изучить основы парадигмы ООП. Реализовать первую часть компьютерной игры на языке программирования C++.

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.

Все части лабораторной работы реализованы в виде классов.

Программа не включает графический интерфейс (пока что), поле выглядит как набор символов, выводящихся в консоль.

- рамка или непроходимая клетка

<пробел> - пустая клетка

> - старт

< - финиш

Базовое поле — пустое поле, ограниченное рамкой, содержит точки входа и выхода, а также непроходимые клетки внутри рамки.

Поле выглядит как рамка, состоящая из символов #, в конце каждой строки находится символ переноса строки.

Например:
#######(\n)
#(\n)
#(\n)
#(\n)
#######(\n)

Класс интерфейса клетки.

Class Icell. Класс содержит чистые виртуальные функции, которые реализует класс клетки. Методы: virtual void change_content(char c) = 0; virtual char get_content() = 0; virtual bool IsEmpty() = 0.

Класс клетки.

Class Cell. Поля: *char content* — символ, содержимое клетки (например, если клетка пустая, то *content = ` `*); *char state* — состояние клетки (состояния описаны в перечислении *states* в файле *states_of_cell.h*, например, если клетка пустая, то поле *state = `e`* и т. д.). Методы: *Cell (char data)* — конструктор, заполняет поле *content* переданным значением data и устанавливает значение *state* в зависимости от содержимого клетки; *change_content (char c)* — меняет значение полей *content* (заполняет переданным *c*) и *state* (в зависимостиот *content*); *get_content ()* - возвращает значение поля *content*; *IsEmpty ()* - возвращает *true*, если *state == empty* (клетка пуста), иначе — *false*.

Класс поля.

Class Field. Поля: *int width* — ширина поля; *int height* — высота поля; *Cell*** *field* — динамический массив клеток. Методы: *Field ()* - конструктор копирования, ширина и высота поля определены директивой *define, width* и *height* заполняются этими значениями, создается массив ссылок на классы клетки, значения для всех клеток передаются *`#`*, для последних в ряду - *`\n`*; *Field (const Field& other)* — конструктор копирования; *Field& operator= (const Field& other)* — оператор копирования; *Field (Field&& other)* — конструктор перемещения; *Field& operator= (Field&& other)* — оператор перемещения; *~Field()* - деструктор, очищается память, диамически выделенная под массив клеток; *Cell*** *get_field()* - возвращает массив клеток; *int get_width()* - возвращает ширину поля; *int get_height()* - возвращает высоту поля; *void print_field()* - печать поля на экран.

Класс итератора для поля.

Class IteratorOfField. Поля: *int ind1* — текущая «игрек-координата» поля; *int ind2* — текущая «икс-координата» поля; *Field* iter_field* — указатель на объект поля, по которому итерируемся; *Cell* iter_cell* — указатель на текущую клетку поля. Методы: explicit IteratorOfField(Field* iter)

— принимает указатель на объект поля, этим значением инициализируется *iter_field*, *ind1* и *ind2* равны 0 (первая клетка в массиве клеток), *iter_cell* — первая клетка массива клеток; *void First()* — устанавливает *iter_cell* как первую клетку массива клеток; *void Next()* — меняет значение *iter_cell*, становится следующей клеткой массива справа, если справа нет, то первой на следующей строке, если клеток нет, то *nullptr*; *bool IsDone()* — возвращает *true*, если все клетки массива клеток обработаны, иначе — *false*; *Cell* Current()* — возвращает *iter_cell; bool Is_Board()* — возвращает *true*, если клетка является граничной или переносом строки; *bool Is_St()* — возвращает *true*, если клетка может быть стартовой клеткой (требуется для генерации клетки старта в строителе), клетка может быть стартовой, если это первый столбец после левой границы; bool Is_Fn() — возвращает *true*, если клетка может быть финишной клеткой (требуется для генерации клетки финиша в строителе), клетка может быть финишной, если это последний столбец перед правой границей; *bool Is_Even()* — возвращает *true*, если клетка пуста и доступна для размещения непроходимых клеток при генерации поля (мной задана закономерность: номер столбца при делении на 3 дает остаток или 1, или 0, а номер строки не кратен 2); *int ret_y()* — возвращает координату по *y*; *int ret_x()* — возвращает координату по *x*.

Класс строителя.

Class Builder. Содержит только чистые виртуальные функции, которые реализуют конкретные строители (в данном случае - строитель базового/стартового/начального). Методы: *virtual void Build_Base_Field(Field& f) = 0*.

Класс строителя базового поля.

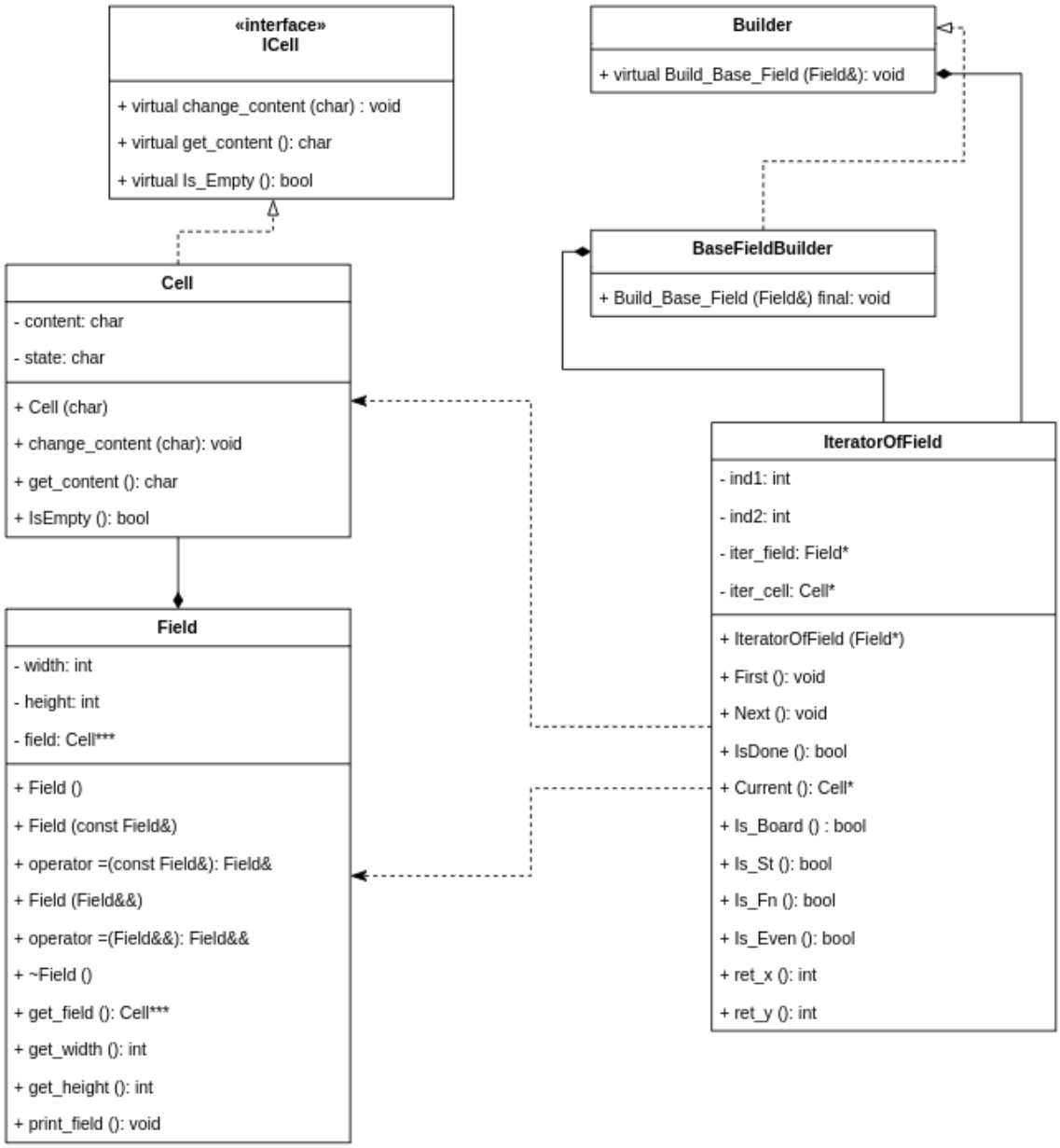
Class BaseFieldBuilder. Наследник класса строителя, реализует его методы. Полей нет. Методы: void Build_Base_Field(Field& field) final — аргумент — ссылка на объект класса поля, заполняет поле внутри пробелами

(кроме рамок и переносов строки), расставляет старт, финиш и непроходимые клетки, используя итератор.

Функция *main()* на данный момент создает объект поля, заполняет его используя объект базового строителя, выводит поле в консоль.

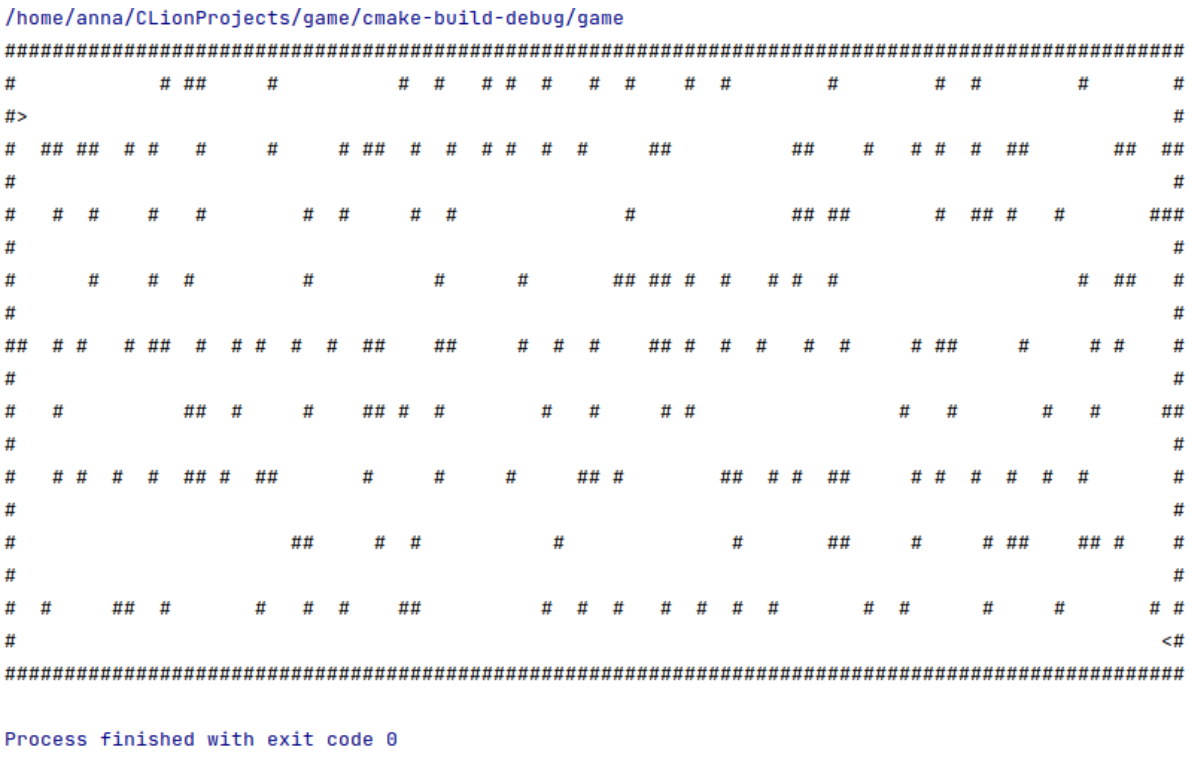
Если в дальнейшем любой из классов будет изменен, это будет отображено в отчете.

UML-диаграмма.



Тестирование.

По итогу работы программы получено поле, ограниченное рамками, заполненное непроходимыми клетками, клетками финиша и старта.



Выводы.

Были изучены основные принципы ООП.

Была реализована первая часть игры, в результате которой выводится базовое поле, готовое к началу игры.