**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 классов**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 0382 |  | Андрющенко К.С |
| Преподаватель |  | Жангиров Т. Р |

Санкт-Петербург

2021

**Цель работы.**

Создание класса игрового поля, его конструкторов и полей, разбитого на клетки, определяемые отдельным классом.

**Задание.**

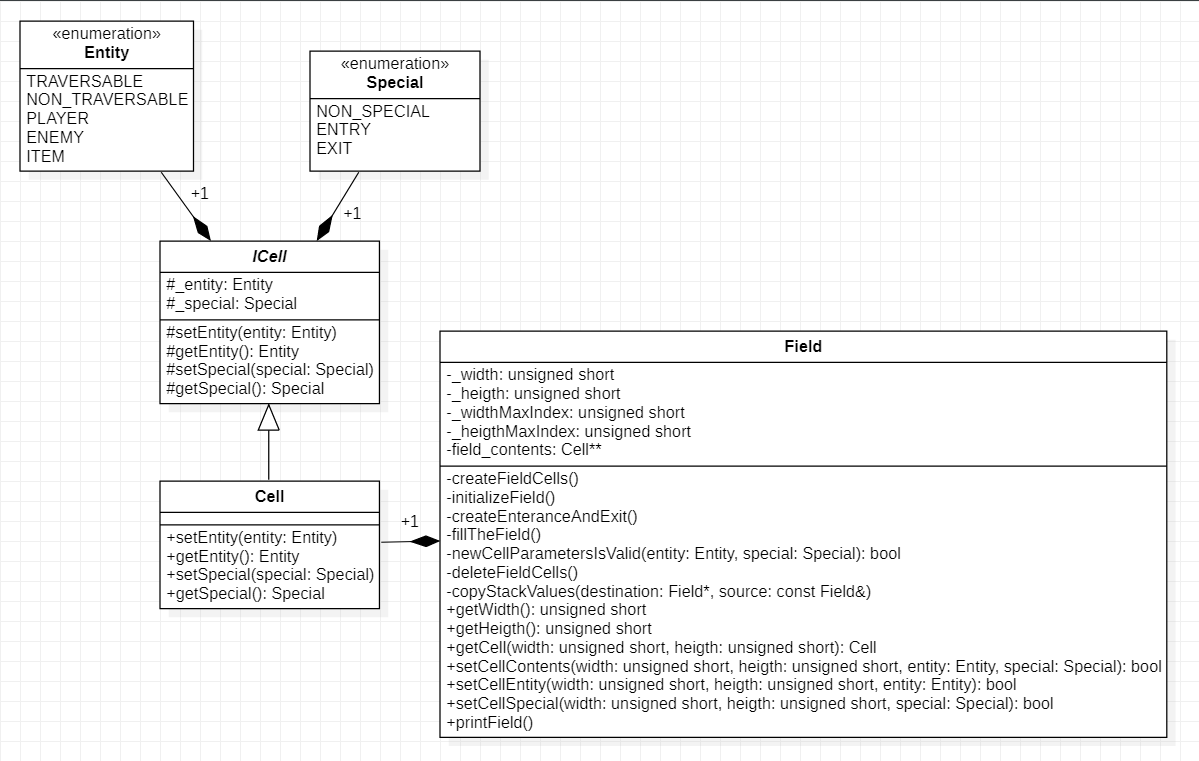
Игровое поле представляет из себя прямоугольную плоскость, разбитую на клетки. На поле на клетках в дальнейшем будут располагаться игрок, враги, элементы взаимодействия. Клетка может быть проходимой или непроходимой, в случае непроходимой клетки, на ней ничего не может располагаться. На поле должны быть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

## 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.

**Выполнение работы.**

UML-диаграмма для данной лабораторной работы см. Рисунок 1.

 Рисунок 1 – UML-диаграмма лабораторной работы №1

Описание диаграммы.

Перечисление Entity хранит информацию о состоянии клетки.

Перечисление Special хранит информацию о том, что находится на клетке.

Отношение между абстрактным классом ICell и перечислениями – композиция, т.к. перечисления являются частью данного класса и не имеют смысла вне его.

Отношение между классом Cell и абстрактным классом ICell – реализация. Cell реализует поведение, заданное поставщиком ICell.

Отношение между классом Cell и Field - композиция, т.к. клетки являются частью класса Field (контейнер) и не имеют смысла вне его. Если поле будет уничтожено, то его содержимое тоже.

Описание классов.

ICell – абстрактный класс, свойства которого – состояние клетки \_entity и содержимое \_special, методы – виртуальные функции, отвечающие за возвращение и установку полей клетки \_special и \_entity. Заголовочный файл см. ПРИЛОЖЕНИЕ A.

Класс клетки Cell является наследником класса ICell. В данном классе переопределены функции возвращающие поля \_special и \_entity и устанавливающие их. Создан конструктор для клетки с неопределенными параметрами, который создает клетку с полями \_entity = Entity::TRAVERSABLE и \_special = Special::NON\_SPECIAL. И конструктор с определенными параметрами \_special и \_entity. Заголовочный файл см. ПРИЛОЖЕНИЕ B.

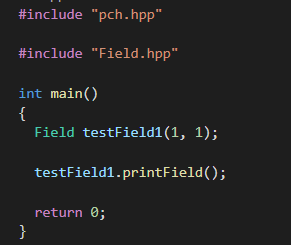
Для класса поля Field были реализованы конструкторы и операторы копирования и перемещения. Методы класса для получения информации о клетке и работе с ней. Методы, реализующие хранение клеток поля, вывод поля на экран, проверка пользовательских данных для построения поля. Метод, при запуске которого происходит подготовка поля к игре. Заголовочный файл см. ПРИЛОЖЕНИЕ C. При передачи неверных параметров в конструктор поля, создается поле размером 2х2, для того чтобы клетки входа и выхода не находились рядом. При переопределении параметров клетки происходит проверка на их соответствие. Сначала создается поле «пустых» клеток необходимых размеров, а затем, установка параметров клеток. При вызове деструктора происходит удаление созданных указателей, что предотвращает утечку памяти. К свойствам поля относятся поля, хранящие его размер.

Проверка классов.

Приложим скриншоты тестов, находящихся в main(), результат и при необходимости памяти после выполнения тестов.

Создание поля неподходящих размеров см.Рисунок 2:

Main():



Output:

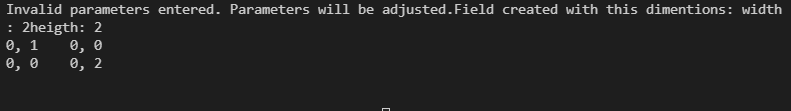
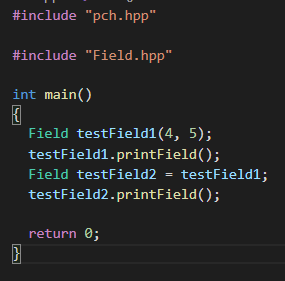


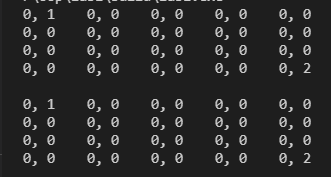
Рисунок - Создание поля неподходящих размеров

Копирование поля см. Рисунок 3:

Main():



Output:



Проверка, что объекты являются разными, но хранят одинаковые значения в полях.

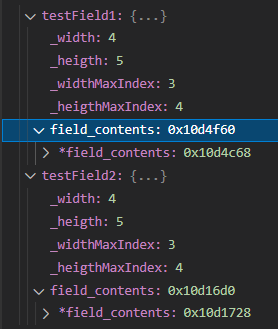
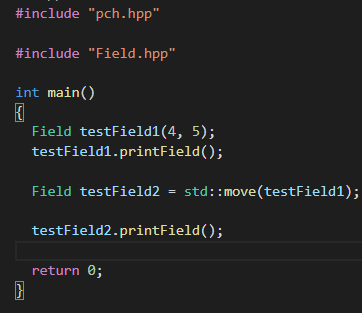


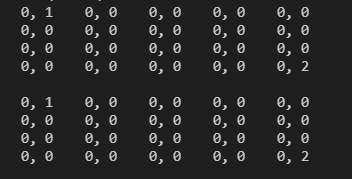
Рисунок - Копирование поля

Перемещение поля см. Рисунок 4:

Main():



Output:



Проверка, что перемещаемы объект остался пустым:

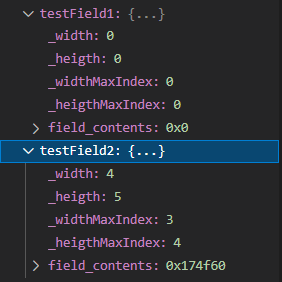
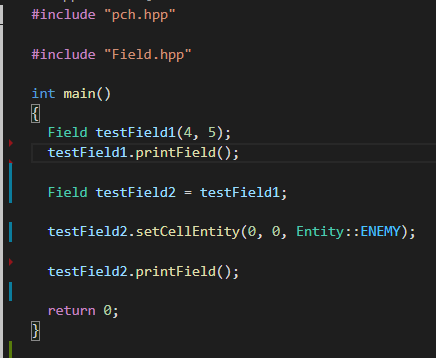


Рисунок - Перемещение поля

Установка значений \_special и \_entity в клетку с координатами width и heigth см. Рисунок 5:

Main():



Output:

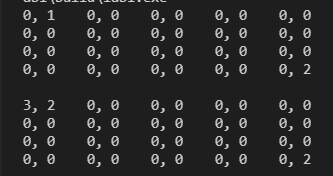


Рисунок - Установка значений \_special и \_entity в клетку с координатами width и heigth

**Вывод.**

В ходе работы были написаны классы поля и клетки.

ПРИЛОЖЕНИЕ A

**ИСХОДНЫЙ КОД ПРОГРАММЫ**

**Файл ICell.hpp**

#pragma once

class ICell

{

protected:

Entity \_entity; //состояние клетки

Special \_special; //что находится на клетке

virtual void setEntity(Entity entity) = 0;

virtual Entity getEntity() = 0;

virtual void setSpecial(Special special) = 0;

virtual Special getSpecial() = 0;

};

ПРИЛОЖЕНИЕ B **Исходный код программ**

**Файл Cell.hpp**

#pragma once

#include "ICell.hpp"

class Cell: public ICell

{

public:

  Cell();

  Cell(Entity entity, Special special);

  Cell(const Cell&) = default; //конструктор копирования. defaul - использование конструктора по умолчанию

  Cell(Cell&&) = default; // конструктор перемещения

  Cell& operator=(const Cell&) = default; //оператор копирования

  Cell& operator=(Cell&&) = default; //оператор перемещения

  ~Cell();

  /// @brief Устанавливает параметр клетки entity.

  void setEntity(Entity entity) override;

  /// @brief Возвращает параметр клетки entity.

  Entity getEntity() override;

  /// @brief Устанавливает параметр клетки special.

  void setSpecial(Special special) override;

  /// @brief Возвращает параметр клетки special.

  Special getSpecial() override;

};

ПРИЛОЖЕНИЕ C **Исходный код программ**

**Файл Field.hpp**

#pragma once

#include "Cell.hpp"

class Field

{

public:

  Field(unsigned short width, unsigned short length);

  Field(const Field& other);

  Field(Field&& other);

  Field& operator=(const Field& other);

  Field& operator=(Field&& other);

  ~Field();

  /// @brief Возвращает параметр клетки width.

  unsigned short getWidth();

  /// @brief Возвращает параметр клетки heigth.

  unsigned short getHeigth();

  /// @brief Возвращают клетку с указанными параметрами.

  Cell getCell(unsigned short width, unsigned short heigth);

  /// @brief Устанавливает параметры клетки.

  /// @param[in] width Кордината клетки по оси абсцисс.

  /// @param[in] heigth Кордината клетки по оси ординат.

  /// @param[in] entity Определяет, что находится в клетке.

  /// @param[in] special Определяет, является ли клетка входом, выходом или ни тем, ни другим.

  /// @return true если клетка существует, false если нет.

  bool setCellContents(unsigned short width, unsigned short heigth, Entity entity, Special special);

  bool setCellEntity(unsigned short width, unsigned short heigth, Entity entity);

  bool setCellSpecial(unsigned short width, unsigned short heigth, Special special);

  /// @brief Вывод поля на экран.

  void printField();

private:

  /// @brief Ширина поля.

  unsigned short \_width = 0;

  /// @brief Длина поля.

  unsigned short \_heigth = 0;

  unsigned short \_widthMaxIndex = 0;

  unsigned short \_heigthMaxIndex = 0;

  /// @brief Указатель на указатель для хранения клеток поля.

  Cell\*\*field\_contents  = nullptr;

  /// @brief Создание клеток поля.

  void createFieldCells();

  /// @brief Запускает функции, подготавливающие поле к игре (createEnteranceAndExit() и fillTheField()).

  void initializeField();

  /// @brief Создаёт клетки выхода и входа на поле. Вход находится на координатах [0][0],

  /// выход на координатах [width][heigth]. Устанавливает игрока в поле входа.

  void createEnteranceAndExit();

  /// @brief Заполняет поле случайными элементами. Для этого используются случайные числа.

  void fillTheField();

  /// @brief Проверка параметров клетки.

  bool newCellParametersIsValid(Entity entity, Special special);

  /// @brief Удаление клеток поля.

  void deleteFieldCells();

  /// @brief Копирование параметров клетки.

  void copyStackValues(Field\* destination, const Field& source);

};