**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №1**

**по дисциплине «Объектно-ориентированное программирование»**

Тема: Создание классов, конструкторов классов, методов классов; наследование

|  |  |  |
| --- | --- | --- |
| Студент гр. 8381 |  | Почаев Н.А. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2020

## Цель работы.

Разработать и реализовать класс игрового поля, набор классов юнитов.

## Задание.

**Основные требования.**

Разработать и реализовать набор классов:

* Класс игрового поля
* Набор классов юнитов

Игровое поле является контейнером для объектов представляющим прямоугольную сетку. Основные требования к классу игрового поля:

* Создание поля произвольного размера
* Контроль максимального количества объектов на поле
* Возможность добавления и удаления объектов на поле
* Возможность копирования поля (включая объекты на нем)
* Для хранения запрещается использовать контейнеры из stl

Юнит является объектов, размещаемым на поля боя. Один юнит представляет собой отряд. Основные требования к классам юнитов:

* Все юниты должны иметь как минимум один общий интерфейс
* Реализованы 3 типа юнитов (например, пехота, лучники, конница)
* Реализованы 2 вида юнитов для каждого типа(например, для пехоты могут быть созданы мечники и копейщики)
* Юниты имеют характеристики, отражающие их основные атрибуты, такие как здоровье, броня, атака.
* Юнит имеет возможность перемещаться по карте

**Дополнительные требования.**

* Созданы конструкторы копирования и перемещения.
* Все методы принимают параметры оптимальным образом (то есть, отсутствует лишнее копирование объектов).
* Для атрибутов юнитов созданы свои классы. Создавать их требуется, если это не противоречит логике.
* Для создания юнитов используются паттерны “Фабричный метод” / “Абстрактная фабрика”.
* Создан итератор для поля.

## Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки CLion, для компиляции и отладки использовалась UNIX-подобная среда Cygwi. Были задействованы пакеты GCC, CMake, а также GDB.

## Реализованные классы

Классы, добавленные в программу в данной лабораторной работе и их функционал представлены в табл. 1. В ней приведено общее описание классов, отдельные моменты пояснены в комментариях к коду.

Таблица 1 – Основные добавленные классы

|  |  |
| --- | --- |
| **Класс** | **Назначение** |
| Vector | Местоположение: ./Auxiliary functionality  Собственная реализация вектора, по функциональности аналогичная std::vector. |
| Array2D  (шаблонный вспомогательный класс двумерного массива) | Местоположение: ./Auxiliary functionality  Класс реализует обёртку над Vector для работы с ним как с двумерным массивом. За счёт этого достигается высокая скорость работы (данные хранятся в памяти последовательно) вкупе с удобством использования методов вектора. |
| GameField | Класс поля. Реализуется хранение в себе юнитов и баз, является посредником при вызове атаки одним юнитом другого, а также отвечает за перемещение юнита (метод вызывается по запросу юнита).  Для данного класса реализовано правило 5-ти: т.е. созданы: пользовательский деструктор, а также конструктор и оператор присвоения семантики копирования и перемещения.  Аналогично для данного класса реализован Input iterator. Со второй лабы является deprecated, т.е. был выполнен переход на хранение ссылок на Cell (описан ниже) не по raw pointer, а по std::shared\_ptr, соответсвенное не поддерживающих необходимые операторы, такие как, например, инкремент ++. |
| Cell | Класс клетки игрового поля. Выполняет хранение в себе указателей на текущий юнит и базу. Является посредником между классом поля и юнитом, базой: даёт доступ только к их методам описания и описанным изменениям характеристик.  Для данного класса также реализовано правило 5-ти. |
| Unit | Местоположение: ./Units  Абстрактный класс юнита. Описывает основные параметры юнитов и методы доступа к ним и их изменения. |
| ObjectFactory | Шаблонная абстрактная Фабрика. Позволяет задавать любой тип идентификатора и любой тип объекта (используется шаблон шаблонов). Позволяет генерировать код фабрик для совершенно разных базовых классов.  Также в данном классе используется паттерн Стратегия для задания политики фабрики – в данном случае возвращение 0 в непредвиденных ситуациях (игнорирование ошибок). |
| ICannonFodder  +  ICavalry  +  IInfantry  +  IShooter  +  IUnitAttack  +  IWizard | Интерфейсы, содержащие в себе специфичные для каждого типа юнитов. Например для типа Wizard объявляются методы magicFist() и healing(), которых нет у других типов. |

В целях сокращения места на описание однотипных классов в таблицу не включены классы по местоположение ./Units/Creatures, т.к. содержат в себе определение интерфейсов, описанных ранее, а также задачу характеристик в конструкторах.

## Выводы.

В ходе выполнения лабораторной работы были написаны требуемые классы поля с сопутствующими классами для хранения данных, а также абстрактный класс юнита с соответствующими производными классами.

# Приложение А Исходный код программы. MAIN.cPP

#include <iostream>

#include "Tests/examples.h"

int main()

{

// fieldBasedTest();

// ObserverDeathTest();

// landscapeTest();

unitInteractionTest();

return 0;

}