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

ОТЧЕТ

по практической работе №1

по дисциплине «Программирование»

ТЕМА: ВВЕДЕНИЕ В ООП.
Создание классов, конструкторов классов, методов классов; наследование.

СТУДЕНТ ГР.0324 _____ КОШЕЛЯЕВ А.С

ПРЕПОДАВАТЕЛЬ _____ ГЛУЩЕНКО А.Г

САНКТ-ПЕТЕРБУРГ

2021

Цель работы.

Получение практических навыков по созданию классов, конструкторов классов, методов классов, паттернов проектирования, наследования.

Основные теоретические положения.

Класс является абстрактным типом данных, определяемым пользователем, и представляет собой модель реального объекта в виде данных и функций для работы с ними.

Данные класса называются полями (по аналогии с полями структуры), а функции класса — методами. Поля и методы называются элементами класса.

Поля класса:

• могут иметь любой тип, кроме типа этого же класса (но могут быть указателями или ссылками на этот класс);

• могут быть описаны с модификатором const, при этом они инициализируются только один раз (с помощью конструктора) и не могут изменяться;

• могут быть описаны с модификатором static, но не как auto, extern и register.

Классы могут быть глобальными (объявленными вне любого блока) и локальными (объявленными внутри блока, например, функции или другого класса).

Особенности локального класса:

• внутри локального класса можно использовать типы, статические (static) и внешние (extern) переменные, внешние функции и элементы перечислений из области, в которой он описан; запрещается использовать автоматические переменные из этой области;

• локальный класс не может иметь статических элементов;

• методы этого класса могут быть описаны только внутри класса;

• если один класс вложен в другой класс, они не имеют каких-либо особых прав доступа к элементам друг друга и могут обращаться к ним только по общим правилам.

Иногда желательно иметь непосредственный доступ извне к скрытым полям класса. Для этого служат дружественные функции и дружественные классы.

Дружественные функции применяются для доступа к скрытым полям класса и представляют собой альтернативу методам. Метод, как правило, используется для реализации свойств объекта, а в виде дружественных функций оформляются действия, не представляющие свойства класса, но концептуально входящие в его интерфейс и нуждающиеся в доступе к его скрытым полям, например, переопределенные операции вывода объектов.

Если все методы какого-либо класса должны иметь доступ к скрытым полям другого, весь класс объявляется дружественным с помощью ключевого слова friend.

Постановка задачи.

Разработать и реализовать набор классов:

Класс игровое поля

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

1. Создание поля произвольного размера.
2. Контроль максимального количества объектов на поле.
3. Возможность добавления и удаления объектов на поле.
4. Возможность копирования поля (включая объекты на нем).
5. Создан итератор поля.

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

1. Все юниты должны иметь как минимум один общий интерфейс.
2. Реализованы 3 типа юнитов (пехота, лучники, конница).

3. Реализованы 2 вида юнитов для каждого типа (Пехота (мечники, копейщик) Лучники (снайпер, скоростной стрелок) Конница (атакующая, оборонная).
4. Юниты имеют характеристики, отражающие их основные атрибуты, такие как здоровье, броня, атака.
5. Юнит имеет возможность перемещаться по карте.

Выполнение работы.

Класс игровоеполе в своей реализации содержит основные методы, заявленные в требованиях к классу.

Создание поля произвольного размера.

```
Gamefield b(15,23);            //создание объекта в стеке
```

Конструктор создаёт объекты класса игровое поле произвольного размера, но не меньше, чем (2*2).

Для примера создал пустое без объектов игровоеполе размера (15*23).

Контроль максимального количества объектов на поле.

Выполняет функция MaxControl(); вызываемая пот необходимости.

Возможность добавления и удаления объектов на поле а так же инициализацию поля выполняет функция DefinitionField(); добавлю объект на поле с координатами (7*3) и выведу поле в консоль.

```
Construktor class Gamefield
Максимально возможное кол-во объектов на поле: (345)
Введите кол-во объектов, которое требуется создать:1
Задайте координату по оси X (строки).
7
Задайте координату по оси Y (столбцы).
3
# # # # # # # # # # # # # # # # # # # # # # # # # # # #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
#       @                                               #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
#                                                       #
# # # # # # # # # # # # # # # # # # # # # # # # # # # # #
```

Рисунок 1. Инициализация поля.

Функция позволяет задавать количество объектов, ограниченное максимально возможным количеством (заданное как один объект одна ячейка). Для примера задам еще пару. (10*10), (14*7).

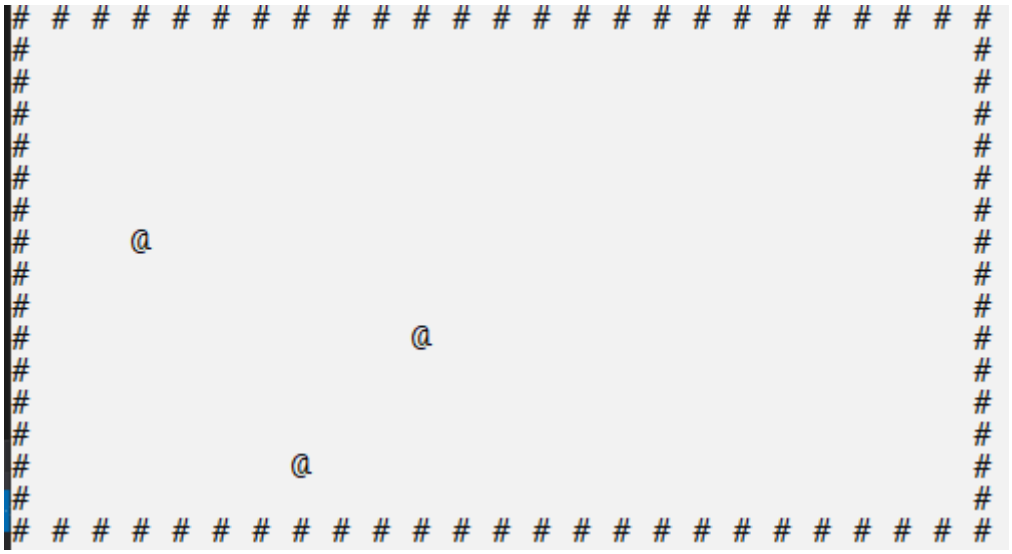

Рисунок 2. 3 объекта на поле.

Возможность копирования поля (включая объекты на нем). Выполняет конструктор копирования.

Между юнитом и полем существует класс объект, который имеет поля координаты X, Y и графического изображения (у каждого объекта на поле свое графическое изображение) у юнита U.

Класс юнит наследуется от класса объект.

Свои поля у юнита — это броня, здоровье, атака, и сумма бойцов так как это отряд.

Создадим юнита через конструктор он создастся в точке (1*2) и выведем в консоль в месте с абстрактными объектами.

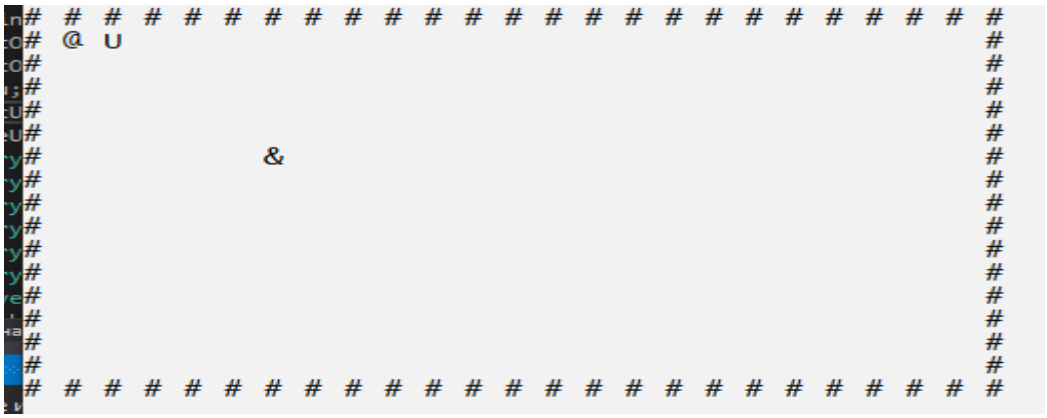

Рисунок 3. Объекты и юнит.

Также реализовано 3 типа юнитов разбитых на 2 вида.

Через класс войн, не содержащий полей, но имеющий виртуальные методы.

Создание воинов выполнено через класс фабричный метод.

```
FactoryAttacking* testikAttacking = new FactoryAttacking;
	FactorySwordsmen* testikSwordsmen = new FactorySwordsmen;
	FactorySpearmen* testikSpearmen = new FactorySpearmen;
	FactorySniper* testukSniper = new FactorySniper;
	FactorySpeed* testtikSpeed = new FactorySpeed;
	FactoryDefense* testDefense = new FactoryDefense;
	std::vector <Warror*> FacPush;
	FacPush.push_back(testikAttacking->create());
	FacPush.push_back(testikSwordsmen->create());
	FacPush.push_back(testikSpearmen->create());
	FacPush.push_back(testukSniper->create());
	FacPush.push_back(testtikSpeed->create());
	FacPush.push_back(testDefense->create());
```

Юнит имеет возможность перемещаться по карте.

Так как рисунки статичны для примера создадим хвост перемещения юнита по карте.

Также к существующим объектам добавим объект с координатами (5*5).

```
a1.SetOneObject(a2, b, 5, 5, '&'); // вот этой функцией.
```

Перемещение по карте осуществляется кнопками (w-вверх, s-вниз, a-в лево, d-в право, p -выход (прервать игру). Опустимся вниз на 4 позиции и повернём на право на 3 позиции.

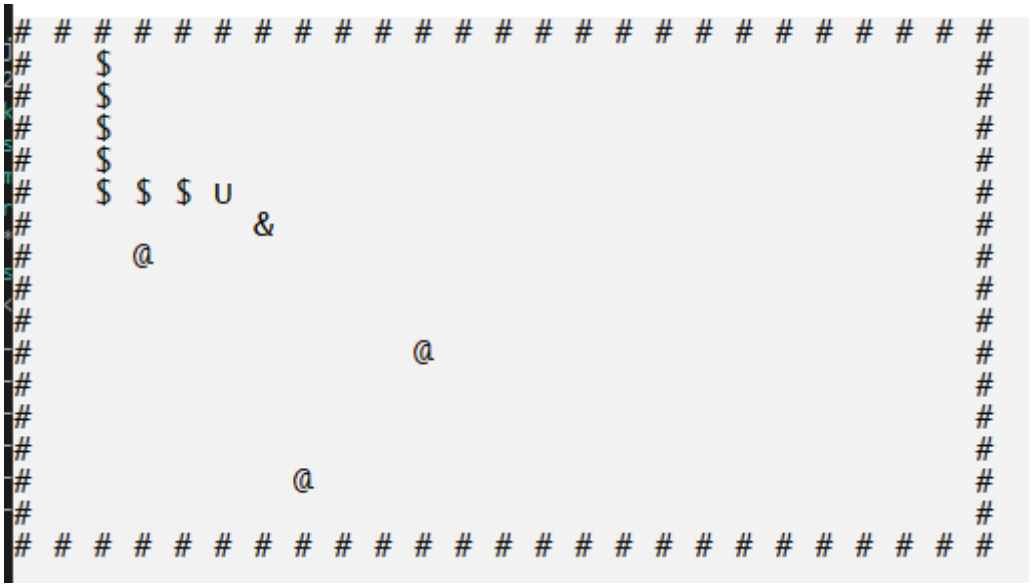

Рисунок 4. Движение отряда по карте.

Нападем на объект (7*3) и уничтожим его.

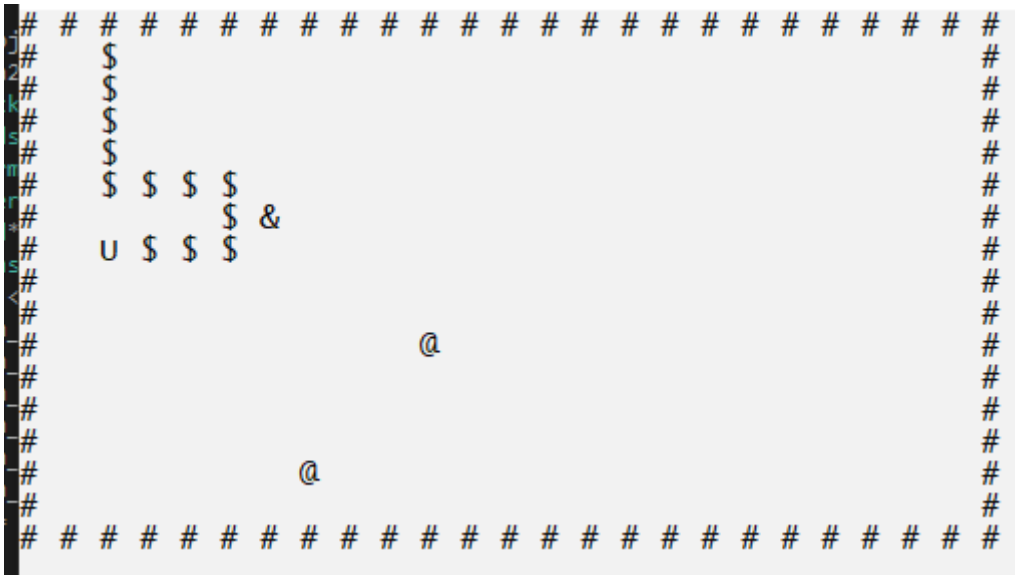

Рисунок 5. Убили объект (7*3)

Выход и оповещение о создании конструктором воинов разных видов, описанных выше.

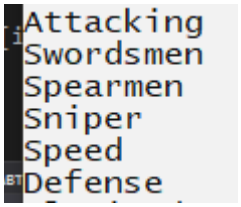

Рисунок 6. Создание воинов.

Для простоты понимания структуры работы программы на Рис. 7 представлена в общем виде на диаграмме классов.

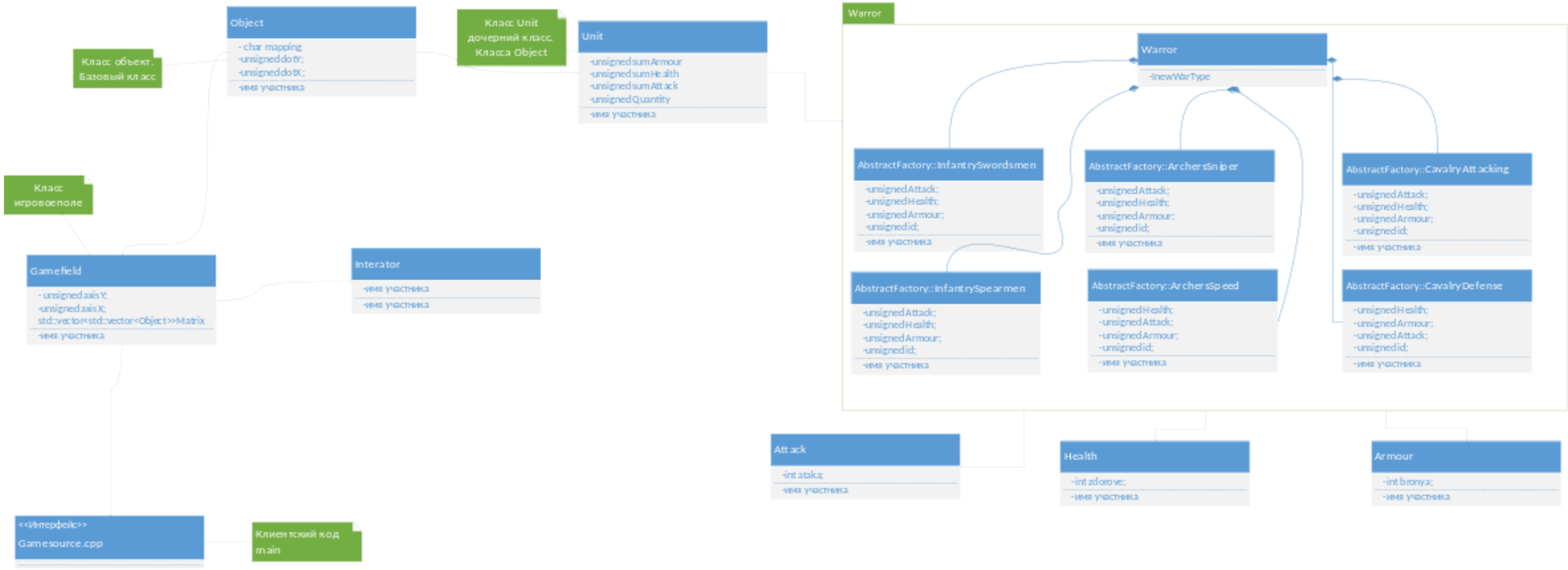

Рисунок 7 Структура классов игры для 1 практической.

Выводы.

В процессе выполнения первой практической работы познакомился с созданием классов используя инкапсуляцию наследование и ООП в части применения паттернов проектирования.

ПРИЛОЖЕНИЕ А

ПОЛНЫЙ КОД ПРОГРАММЫ

Код программы