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

ОТЧЕТ

по практической работе № 4

по дисциплине «ООП»

Тема: Полиморфизм

Студент гр. 8383 _____ Степанов В.Д.

Преподаватель _____ Жангиров Т. Р.

Санкт-Петербург

2020

Цель работы.

Изучение и применение паттернов программирования для разработки игры на языке C++.

Постановка задачи.

Реализовать набор классов, для ведения логирования действий и состояний программы. Основные требования:

- Логирование действий пользователя
- Логирование действий юнитов и базы

Выполнены основные требования к логированию
Реализована возможность записи логов в файл
Реализована возможность записи логов в терминал
Взаимодействие с файлами должны быть по идиоме RAII
*Для логирования состояний перегружен оператор вывода в поток
*Переключение между разным логированием (логирование в файл, в терминал, без логирования) реализуется при помощи паттерна “Прокси”
*Реализован разный формат записи при помощи паттерна “Адаптер”

Ход работы.

1. Добавлена возможность записи логов файл, в терминал или вообще не выводить. Выбор предоставляется сделать при запуске программы. В случае если выбрана запись в файл, то легирование будет записываться в файл под названием “LogFile.txt”. Реализация находится в файле Game.cpp в методе startGame.
2. Взаимодействие с файлом записи логово реализовано по идиоме RAII. Реализация находится в файлах ProxyLog.cpp и ProxyLog.hpp.
3. Выбор реализации логирования реализован при помощи паттерн “Прокси”. Записывать логи могут class Unit (удар другого юнита), class Base (создание и удаление юнита), class MoveCommand (перемещение юнита,

взаимодействие с разными ландшафтами) и hitCommand (атака юнита, взаимодействие юнита с нейтральными объектами). Реализация находится в файлах ProxyLog.cpp и ProxyLog.hpp.

4. Реализован разный формат записи при помощи паттерна “Адаптер”. Каждый из классов, имеющих возможность записывать логи, используют методы класса Adapter, чтобы представить информацию для записи логов. Реализация находится в файлах Adapter.cpp и Adapter.hpp.

Примечание: Перегрузка оператора вывода в поток не реализована.

Пример работы программы.

Запустим программу. Начальная карта представлена на рисунке 1.

```
Запись логов в файл 'f' или в консоль 'c'. Без записи  'n'
f
File
Start
 --------------
|  BA    <1  B1|
|HP  ##!5  $1  |
|      WW!3  <3|
|  6!PC  PC!1  |
|2>  4!WW      |
|  2$  2!    BA|
|B2  4>    HP  |
 --------------
$1  h = 15  a = 15  p = 1   2$  h = 15  a = 15  p = 1
<1  h = 5   a = 10  p = 4   2>  h = 5   a = 10  p = 4
<3  h = 5   a = 10  p = 4   4>  h = 5   a = 10  p = 4
!1  h = 8   a = 13  p = 5   2!  h = 8   a = 13  p = 5
!3  h = 8   a = 13  p = 5   4!  h = 8   a = 13  p = 5
!5  h = 8   a = 13  p = 5   6!  h = 8   a = 13  p = 5
```

Рисунок 1 – Начальная карта

Взаимодействуем юнитом “!1” с нейтральным объектом “PC” ^ переместим юнита “!5” влево, на ландшафт “##”, взаимодействуем юнитом “!1” с нейтральным объектом “PC” и атакуем юнитом “$1”. Результат взаимодействий представлен на рисунке 2. Записанные логи представлены на рисунке 3.

```
 --------------
|  BA     <1  B1|
|HP  !5     $1  |
|       WW!3  <3|
|  6!PC  PC!1   |
|2>  4!WW       |
|  2$  2!     BA|
|B2  4>     HP  |
 --------------
$1  h = 15  a = 15  p = 1   2$  h = 15  a = 15  p = 1
<1  h = 4   a = 10  p = 4   2>  h = 5   a = 10  p = 4
<3  h = 4   a = 10  p = 4   4>  h = 5   a = 10  p = 4
!1  h = 6   a = 13  p = 6   2!  h = 8   a = 13  p = 5
!3  h = 7   a = 13  p = 5   4!  h = 8   a = 13  p = 5
!5  h = 7   a = 13  p = 5   6!  h = 8   a = 13  p = 5
```

Рисунок 2 – Результат взаимодействия

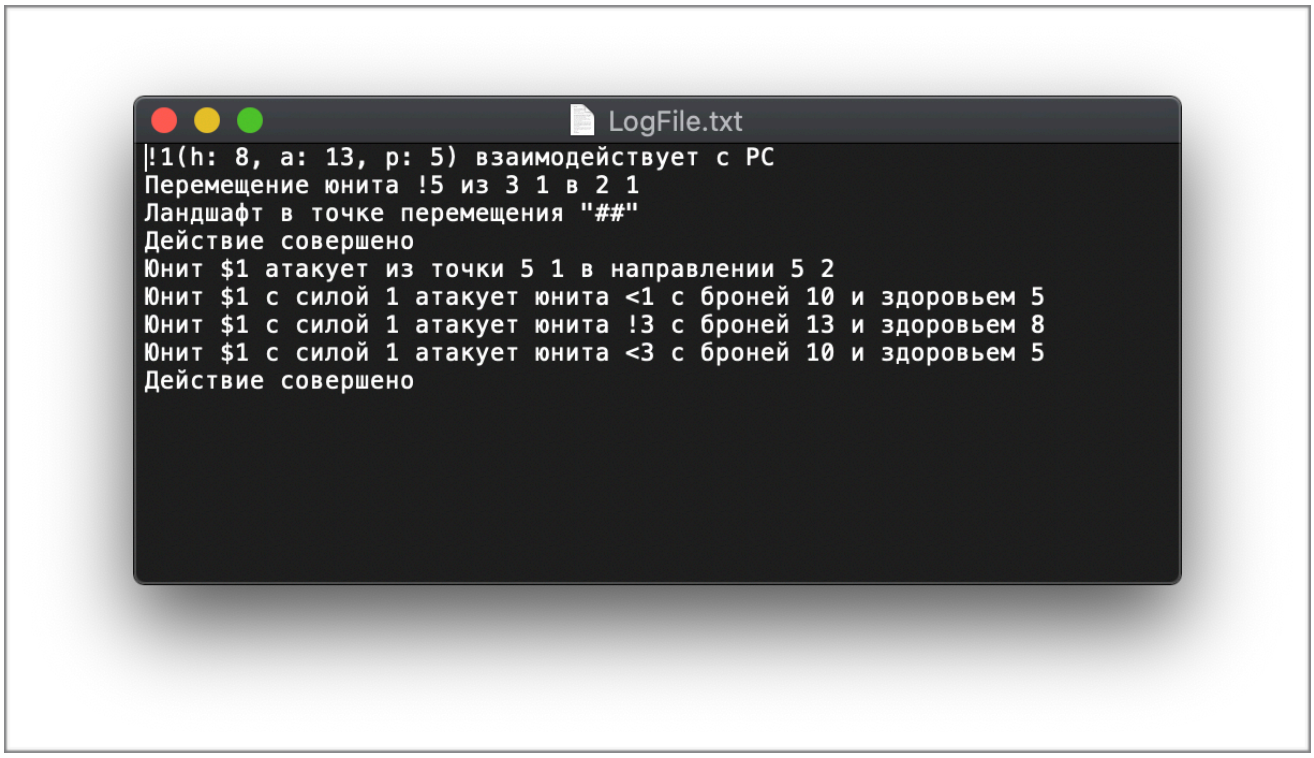

Рисунок 3 – Записанные логи

Вывод.

В ходе лабораторной работы были изучены и применены паттерноы программирования для разработки игры на языке C++.