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

# ОТЧЕТ

**по лабораторной работе №3**
**по дисциплине «Объектно ориентированное программирование»**
**Тема: Логирование, перегрузка операций**

Студент гр. 1304 Шаврин А.П.

Преподаватель Жангиров Т.Р.

Санкт-Петербург
2022

**Цель работы.**

Изучить методы логирования с помощью ООП и перегрузку операций в языке C++, а также применить полученные знания в реализации игры. Научиться выстраивать архитектуру проекта с возможностью дальнейшего расширения.

**Задание.**

Реализовать класс/набор классов отслеживающих изменения состояний в программе. Отслеживание должно быть 3-х уровней:

1. Изменения состояния игрока и поля, а также срабатывание событий
2. Состояние игры (игра начата, завершена, сохранена, и.т.д.)
3. Отслеживание критических состояний и ошибок (поле инициализировано с отрицательными размерами, игрок попытался перейти на непроходимую клетку, и.т.д.)

Реализованы классы для вывода информации разных уровней для в консоль и в файл с перегруженным оператором вывода в поток.

Требования:

- Разработан класс/набор классов отслеживающий изменения разных уровней
- Разработаны классы для вывода в консоль и файл с соблюдением идиомы RAII и перегруженным оператором вывода в поток.
- Разработанные классы спроектированы таким образом, чтобы можно было добавить новый формат вывода без изменения старого кода (например, добавить возможность отправки логов по сети)
- Выбор отслеживаемых уровней логирования должен происходить в runtime
- В runtime должен выбираться способ вывода логов (нет логирования, в консоль, в файл, в консоль и файл)

Примечания:

- Отслеживаемые сущности не должны ничего знать о сущностях, которые их логируют

- Уровни логирования должны быть заданными отдельными классами или перечислением
- Разные уровни в логах должны помечаться своим префиксом
- Рекомендуется сделать класс сообщения
- Для отслеживания изменений можно использовать наблюдателя
- Для вывода сообщений можно использовать адаптер, прокси и декоратор

**Выполнение работы.**

1. Было создано перечисление *LogLevels {Errors, Processes, GameStates},* хранящее в себе уровни логов.
2. Было создано перечисление *LogCout {Console, File},* хранящее в себе типы потоков вывода логов.
3. После был реализован класс *Log,* имеющий 2 поля: *level* типа *LogLevels*; *log_message* типа *std::string*. Конструктор класса принимает 2 аргумента типа *LogLevels* и *std::string*, устанавливая соответсвующие поля класса он добавляет префикс уровня лога в *log_message*. Также данный класс перегружает оператор << вывода в поток.
4. Был реализован класс-интерфейс *Logger,* определяющий общее поведение всех логеров. Данный класс имеет чисто виртуальный метод *print,* принимающий указатель на класс *Log*.
5. От класса *Logger* был унаследован класс *ConsoleLogger*, переопределяющий метод *print,* в котором производится вывод лога в консоль.
6. От класса *Logger* был унаследован класс *FileLogger*, имеющий одно приватное поле *std::ofstream log_file*. Конструктор данного класса принимает название файла (переменную *std::string*), в который будет производиться логирование и открывает данный файл, очищая от предыдущих логов, если он уже был создан, или создавая этот файл. В переопределенном методе *print,* производится вывод лога в файл.
7. После был создан класс *LogController*

Данный класс имеет поля:

- *ConsoleLogger* console_logger;*
- *FileLogger* file_logger;*
- *LogLevels s level*;
- *OutputStreams log_cout*; (вектор из *LogCout*)

Метод *setLevel* получает *LogLevels* и устанавливает значение поля класса в соответствующее значение.

Метод *addStream* получает *LogCout* и добавляет поток для вывода логов, если его еще нет в *log_cout*.

Метод *removeStream* получает *LogCout* и проверяют, есть ли среди отслеживаемых потоков удаляемый элемент, если да, то удаляет его.

Метод *userDialog*, осуществляет диалог с пользователем и заполняет поля *level* и *log_cout*, путем вызова методов *ConvertLevel* и *ConvertStreams*, которые конвертируют введенные пользователем значения в значения *LogLevels* и *LogCout* и заносят их в поля класса с помощью методов *setLevel* и *addStream*.

В методе *setParametrs* производится вызов *userDialog*, пока пользователь не введет параметры правильно или пока не откажется от изменений.

Метод *handleLog*, принимающий указатель на *Log*, осуществляет проверку надобности вывода лога и потоки вывода.

8. Затем был модернизован класс-интерфейс *Mediator*, имеющий чисто виртуальный метод *send*, который позволяет отправить некое сообщение, принимаемое этим методом. Теперь это шаблонный класс, позволяющий отправлять сообщения разных типов.

9. После был модернизован класс *GameElement*, который в своем поле хранит указатель на медиатор, а также имеет метод *setMediator*, позволяющий установить этот медиатор. Теперь это также шаблонный класс, который может хранить медиаторы разного типа.

10. Был реализован конкретный медиатор *LogMediator*, наследующийся от класса интерфейса *Mediator<Log*>* (Медиатор посылающий указатели на класс *Log*), имеющий единственное поле – указатель на

*LogController*. Данный класс переопределяет метод send для родительского класса, в котором вызывает у *LogController* метод *handleLog* и передает туда полученный указатель на класс *Log*.

11. Затем все необходимые классы для логирования были унаследованы от *GameElement<Log*>* и в нужных местах стали передавать логи.

12. Изменение отслеживаемых уровней и потоков в *runtime*, реализовано в *CommandReaderMediator*, соответствующая клавиша либо добавляет уровень или поток, либо удаляет.

Зависимости классов приведены на UML диаграмме ниже:

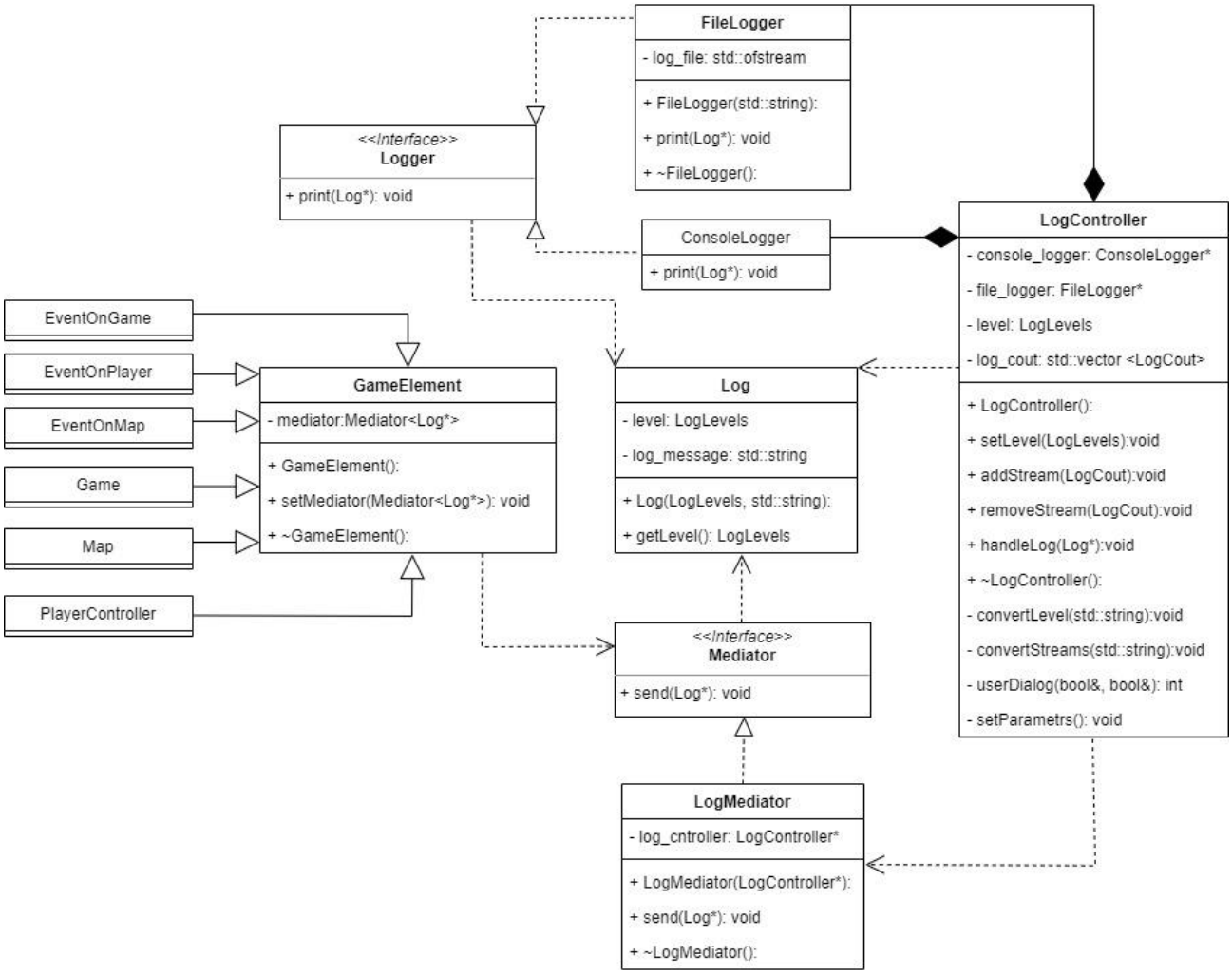



Рисунок 1: UML диаграмма классов.

Результаты тестирования см. в приложении Б.

**Выводы.**

Изучены методы логирования с помощью ООП и перегрузка операций в языке C++. Полученные знания применены в реализации игры. Научились выстраивать архитектуру проекта с возможностью дальнейшего расширения.

# ПРИЛОЖЕНИЕ Б

# ТЕСТИРОВАНИЕ

1. Запуск программы со стартовыми параметрами

```
alex@alex-VirtualBox:~/Leti/OOP/labs_on_folders$ ./sfml-app
Хотите задать свой уровень логирования? (По умолчанию: errors) [y]: n
Хотите задать куда выводить логи? (По умолчанию: console) [y]: n
По ходу программы можно задавать параметры логирования с помощью клавиатуры.
Установить уровень логирования:
0 - errors      1 - processes   2 - game states
Добавить поток логирования:
6 - console     7 - file
Удалить поток логирования:
F6 - console    F7 - processes
Хотите задать свои значения поля (вместо 5 на 5)? [y]: n
Setting vertical sync not supported
LogLevel: 0 ; Message: Error!!! Player tried to hit without energy
LogLevel: 0 ; Message: Error!!! Player tried to move to an impassable cell or an enemy
alex@alex-VirtualBox:~/Leti/OOP/labs_on_folders$
```

2. Проверка возможности изменения стартовых значений

```
alex@alex-VirtualBox:~/Leti/OOP/labs_on_folders$ ./sfml-app
Хотите задать свой уровень логирования? (По умолчанию: errors) [y]: y
0 - errors, 1 - processes, 2 - game_states
2
Хотите задать куда выводить логи? (По умолчанию: console) [y]: y
0 - console, 1 - file
Вводите потоки вывода логов по следующему образцу: <index><index>
01
По ходу программы можно задавать параметры логирования с помощью клавиатуры.
Установить уровень логирования:
0 - errors      1 - processes   2 - game states
Добавить поток логирования:
6 - console     7 - file
Удалить поток логирования:
F6 - console    F7 - processes
Хотите задать свои значения поля (вместо 5 на 5)? [y]: n
Setting vertical sync not supported
LogLevel: 2 ; Message: Game started
LogLevel: 1 ; Message: Map was create
LogLevel: 1 ; Message: Map got a plyer
LogLevel: 1 ; Message: Enemy has been added to the map
LogLevel: 1 ; Message: Enemy has been added to the map
LogLevel: 1 ; Message: Player hit
LogLevel: 1 ; Message: Enemy has been removed from the map
LogLevel: 1 ; Message: Player hit
LogLevel: 1 ; Message: Event (set armor) was execute
LogLevel: 0 ; Message: Error!!! Player tried to move to an impassable cell or an enemy
LogLevel: 1 ; Message: Event (set win game) was execute
LogLevel: 1 ; Message: Event (set energy) was execute
You win!!!
LogLevel: 1 ; Message: Event (win game) was execute
LogLevel: 2 ; Message: Game ended
alex@alex-VirtualBox:~/Leti/OOP/labs_on_folders$
```

Logs.txt M

OOP > labs_on_folders > Logs.txt

```
LogLevel: 2 ; Message: Game started
LogLevel: 1 ; Message: Map was create
LogLevel: 1 ; Message: Map got a plyer
LogLevel: 1 ; Message: Enemy has been added to the map
LogLevel: 1 ; Message: Enemy has been added to the map
LogLevel: 1 ; Message: Player hit
LogLevel: 1 ; Message: Enemy has been removed from the map
LogLevel: 1 ; Message: Player hit
LogLevel: 1 ; Message: Event (set armor) was execute
LogLevel: 0 ; Message: Error!!! Player tried to move to an impassable cell or an enemy
LogLevel: 1 ; Message: Event (set win game) was execute
LogLevel: 1 ; Message: Event (set energy) was execute
LogLevel: 1 ; Message: Event (win game) was execute
LogLevel: 2 ; Message: Game ended
```

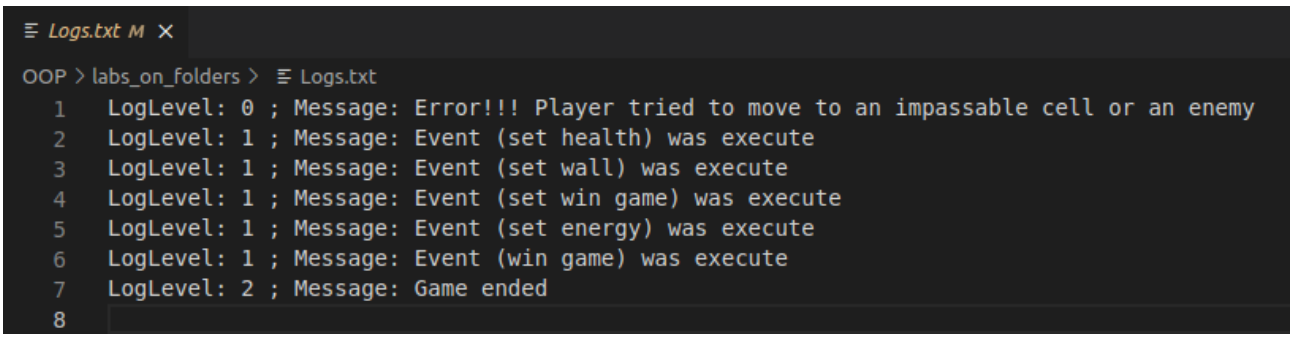
3. Проверка возможности изменения уровней логирования в *runtime*

(По ходу игры был изменен уровень с *errors* на *game states*)

```
alex@alex-VirtualBox:~/Leti/OOP/labs_on_folders$ ./sfml-app
Хотите задать свой уровень логирования? (По умолчанию: errors) [y]: n
Хотите задать куда выводить логи? (По умолчанию: console) [y]: n
По ходу программы можно задавать параметры логирования с помощью клавиатуры.
Установить уровень логирования:
0 - errors      1 - processes   2 - game states
Добавить поток логирования:
6 - console     7 - file
Удалить поток логирования:
F6 - console    F7 - processes
Хотите задать свои значения поля (вместо 5 на 5)? [y]: n
Setting vertical sync not supported
LogLevel: 0 ; Message: Error!!! Player tried to hit without energy
LogLevel: 0 ; Message: Error!!! Player tried to move to an impassable cell or an enemy
LogLevel: 1 ; Message: Event (set health) was execute
You loose!!!
LogLevel: 1 ; Message: Event (end game) was execute
LogLevel: 2 ; Message: Game ended
```

(По ходу игры был изменен поток)

```
alex@alex-VirtualBox:~/Leti/OOP/labs_on_folders$ ./sfml-app
Хотите задать свой уровень логирования? (По умолчанию: errors) [y]: y
0 - errors, 1 - processes, 2 - game_states
2
Хотите задать куда выводить логи? (По умолчанию: console) [y]: n
По ходу программы можно задавать параметры логирования с помощью клавиатуры.
Установить уровень логирования:
0 - errors      1 - processes   2 - game states
Добавить поток логирования:
6 - console     7 - file
Удалить поток логирования:
F6 - console    F7 - processes
Хотите задать свои значения поля (вместо 5 на 5)? [y]: n
Setting vertical sync not supported
LogLevel: 2 ; Message: Game started
LogLevel: 1 ; Message: Map was create
LogLevel: 1 ; Message: Map got a plyer
LogLevel: 1 ; Message: Enemy has been added to the map
LogLevel: 1 ; Message: Enemy has been added to the map
LogLevel: 1 ; Message: Player hit
LogLevel: 1 ; Message: Player taked damage
LogLevel: 1 ; Message: Enemy has been removed from the map
LogLevel: 1 ; Message: Player hit
LogLevel: 1 ; Message: Event (set armor) was execute
LogLevel: 0 ; Message: Error!!! Player tried to hit without energy
You win!!!
```


Logs.txt M
OOP > labs_on_folders > Logs.txt
1 LogLevel: 0 ; Message: Error!!! Player tried to move to an impassable cell or an enemy
2 LogLevel: 1 ; Message: Event (set health) was execute
3 LogLevel: 1 ; Message: Event (set wall) was execute
4 LogLevel: 1 ; Message: Event (set win game) was execute
5 LogLevel: 1 ; Message: Event (set energy) was execute
6 LogLevel: 1 ; Message: Event (win game) was execute
7 LogLevel: 2 ; Message: Game ended
8