# 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.

|  |  |  |
| --- | --- | --- |
| Студент гр. 2381 |  | Двиков Д.В. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

# Цель работы.

Изучить понятия класса, его методов и полей, научиться реализовывать простейшие классы и осуществлять межклассовые отношения.

# Задание.

а) Создать класс игрока. У игрока должны быть поля, которые определяют его характеристики, например кол-во жизней, очков и.т.д. Также в классе игрока необходимо реализовать ряд методов для работы с его характеристиками. Данные методы должны контролировать значения характеристик (делать проверку на диапазон значений).

б) Создать класс, передвигающий игрока по полю и работу с характеристиками. Данный класс всегда должен знать об объекте игрока, которым управляет, но не создавать класс игрока. В следующих лаб. работах данный класс будет проводить проверку, может ли игрок совершить перемещение по карте.

# Примечания:

# Не забывайте для полей и методов определять модификатор доступа

# Для указания направления движения можно использовать перечисление enum или дополнительную систему классов. Использования чисел или строк является для указания направления является плохой практикой

# Делать отдельный метод под каждое направление делает класс перегруженным, и в будущем ограничивает масштабирование класса

# Выполнение работы:

Используется стандартная библиотека c++ и её заголовочный файл *iostream.*

1. Определяется класс *Player*, объектoм которого является игрок.
   1. Реализуются методы класса с модификатором доступа *public*:

* *explicit Player(int health = 100, int score =0);* - конструктор класса *Player.*
* *int getHealth() const; -* возвращает поле health.
* *int getPoints() const; -* возвращает поле points.
* *int getX() const; - возвращает поле x;*
* *int getY() const; - возвращает поле y;*
* *void setHealth(int health); -* меняем значение поля *health* на значение, переданное в функцию.
* *void setScore(int score); -* меняем значение поля *score* на значение, переданное в функцию.
* *void setХ(int x); -* меняем значение поля *points* на значение, переданное в функцию.
* *void setY(int y); -* меняем значение поля *armor* на значение, переданное в функцию.
* *~Player() = default; -* деструктор класса Player.
  1. Инициализируются поля класса с модификатором доступа *private*:
* *int health;* – здоровье игрока.
* *int score;* – очки игрока.
* *int x;* - позиция по х.
* *int y;* - позиция по y.

1. Определяется класс *playercontroller*, отвечающий за управление игрока, а также вывод информации об игроке.
   1. Реализуются методы класса с модификатором доступа *public*:

* *playercontroller(Player& p);* - конструктор класса playercontroller.
* *void stats();* - выводит в консоль значение полей класса Player.
* *void move(Directions direction);* - изменяет координаты игрока.
* void getHealth(int health); - увеличивает значение поля health класса Player.
* void GetScore(int score); - увеличивает значение поля score класса Player.
* void Heal(int health); - увеличивает значение поля health класса Player.
* *~* Controller*() = default; -* деструктор класса Controller.
  1. Инициализируются поля класса с модификатором доступа *private*:
* *Player& player*; - ссылка на объект класса игрока.

**Архитектура программы.**

В *main.cpp* инициализируется объект класса *Controller*. Контроллер начинает игру с помощью метода *StartGame ()*, где, в свою очередь c помощью функций класса запускается игра и выводятся характеристики игрока.

Класс *Player* связан с классом Контроллера с помощью координат игрока, хранящихся в классе *Player* и изменяющихся методом *move ()*.

# Результат работы программы:

# 

Рис 1. – демонстрация работы программы в терминале Powershell

# 

# Выводы.

Были изучены основы объектно-ориентированного программирования. В ходе лабораторной работы были созданы классы, отвечающие за игрока и взаимодействие пользователя с игрой.