**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**по лабораторной работе №1**

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

**Тема: Создание классов**

|  |  |  |
| --- | --- | --- |
| Студент гр. 2303 |  | Ринг С.О. |
| Преподаватель |  | Жангиров Т.Р. |

Санкт-Петербург

2023

**Цель работы.**

Изучение основных принципов объектно-ориентированного программирования. Написание первых классов с реализацией базовой логики нашей будущей игры.

**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.

**Выполнение работы.**

Была создана вспомогательная структура Vector с двумя конструкторами. Первый принимает два аргумента x и y типа int. Второй также принимает пару значений restrictX и пару значений restrictY (ограничение обеих координат сверху и снизу), чтобы при помощи функции clamp ограничить значение полей структуры (с целью, чтобы при перемещении игрока по игровому полю не выйти за границы матрицы). Также переопределены методы operator+ и operator+=, чтобы иметь возможность складывать векторы.

Класс игрока Player. В заголовочном файле Player.h прописаны приватные поля:

* + health - unsigned int поле, отвечающее за здоровье игрока
  + score - unsigned int поле, в котором хранится счет игрока.

Также прописаны сигнатуры функций, а затем реализованы в файле Player.cpp:

* + Конструктор, который принимает соответствующие значения и заполняет каждое поле класса;
  + Геттеры для каждого поля класса - соответствующая функция, позволяющая получить значение приватного поля класса;
  + Сеттеры для каждого поля класса - соответствующая функция, позволяющая установить значение приватного поля класса.

Класс управляющий игроком - PlayerManager. В нем есть поле player, в котором хранится ссылка на объект класса Player и поле coord - координаты игрока. В нем также реализованы методы addHealth и addScore, чтобы иметь возможность поменять значение этих полей в классе Player и методы getCoord, setCoord и changeCoord для операций с координатами игрока.

Функция changeCoord, которая принимает только направление. Чтобы задавать направление реализован вспомогательный enum class Direction, в котором есть 4 варианта перемещения: TOP, DOWN, LEFT, RIGHT. В зависимости от переданного направления изменяется одна из координат на 1 или -1. Допустим, при направлении “вниз” к значению y прибавляется -1.

**Выводы.**

Во время выполнения работы были созданы несколько классов с необходимыми полями и методами на языке программирования C++, а также проведено ознакомление с парадигмами объектно-ориентированного программирования.

**ПРИЛОЖЕНИЕ A.**

**Исходный код программы.**

Файл Vector.h

#ifndef GAME\_VECTOR\_H

#define GAME\_VECTOR\_H

#include <utility>

#include <cinttypes>

#include <algorithm>

struct Vector {

int x;

int y;

Vector(int x, int y);

Vector(int x, int y, const std::pair<int, int> &restrictX, const std::pair<int, int> &restrictY);

Vector operator+(const Vector &other) const;

void operator+=(const Vector &other);

};

#endif //GAME\_VECTOR\_H

Файл Vector.cpp

#include "Vector.h"

void Vector::operator+=(const Vector &other) {

this->operator=(operator+(other));

}

Vector Vector::operator+(const Vector &other) const {

return {(int)(x + other.x), (int)(y + other.y)};

}

Vector::Vector(int x, int y, const std::pair<int, int> &restrictX, const std::pair<int, int> &restrictY) :

Vector(

std::clamp(x, restrictX.first, restrictX.second),

std::clamp(y, restrictY.first, restrictY.second)

) {

}

Vector::Vector(int x, int y) :

x(x), y(y){

}

Файл Direction.h

#ifndef GAME\_DIRECTION\_H

#define GAME\_DIRECTION\_H

enum class Direction{

TOP,

DOWN,

LEFT,

RIGHT

};

#endif //GAME\_DIRECTION\_H

Файл Player.h

#ifndef GAME\_PLAYER\_H

#define GAME\_PLAYER\_H

#include <cinttypes>

#include <algorithm>

class Player {

private:

unsigned int health;

unsigned int score;

public:

Player(unsigned int health, unsigned int score);

unsigned int getHealth() const;

unsigned int getScore() const;

void setHealth(int newHealth);

void setScore(unsigned int newScore);

};

#endif //GAME\_PLAYER\_H

Файл Player.cpp

#include "Player.h"

unsigned int Player::getHealth() const {

return health;

}

unsigned int Player::getScore() const {

return score;

}

void Player::setHealth(int newHealth) {

health = std::clamp(newHealth, 0, 100);

}

void Player::setScore(unsigned int newScore) {

score = newScore;

}

Player::Player(unsigned int health, unsigned int score) :

health(health),

score(score){

}

Файл PlayerManager.h

#ifndef GAME\_PLAYERMANAGER\_H

#define GAME\_PLAYERMANAGER\_H

#include <memory>

#include "Vector.h"

#include "Player.h"

#include "Direction.h"

class PlayerManager {

private:

Player& player;

Vector coord;

public:

PlayerManager(Player& player, const Vector& coord);

void addHealth(int addition);

void addScore(unsigned int addition);

void changeCoord(Direction dir);

void setCoord(const Vector& newCoord);

const Vector& getCoord();

};

#endif //GAME\_PLAYERMANAGER\_H

Файл PlayerManager.cpp

#include "PlayerManager.h"

PlayerManager::PlayerManager(Player& player, const Vector& coord) :

player(player), coord(coord) {

}

void PlayerManager::addHealth(int addition) {

player.setHealth(player.getHealth() + addition);

}

void PlayerManager::addScore(unsigned int addition) {

player.setScore(player.getScore() + addition);

}

const Vector& PlayerManager::getCoord() {

return coord;

}

void PlayerManager::setCoord(const Vector& newCoord) {

coord = newCoord;

}

void PlayerManager::changeCoord(Direction dir) {

Vector newCoord = getCoord();

switch (dir) {

case Direction::TOP:

newCoord += Vector{0, 1};

break;

case Direction::DOWN:

newCoord += Vector{0, -1};

break;

case Direction::LEFT:

newCoord += Vector{-1, 0};

break;

case Direction::RIGHT:

newCoord += Vector{1, 0};

break;

}

setCoord(Vector(newCoord.x, newCoord.y,{0, 100},{0, 100}));

}