Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного обеспечения и администрирования информационных систем

Направление подготовки математическое обеспечение и администрирование информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №2**

«Практическая реализация классов»

Выполнил:

студент группы 213.1 Козявин М. С.

Проверил:

старший преподаватель кафедры ПОиАИС Ураева Е. Е.

Курск, 2022

***Цель работы:*** изучить основные приемы реализации классов на языке C++.

***Задание***

*Задача 1.* В соответствии с базовым заданием разработать класс (или систему классов) и

программу, иллюстрирующую его возможности.

Требования к классам с данными:

− наличие в классе закрытых полей;

− наличие функций доступа (в т.ч. модификации) к закрытым полям класса;

Требования к классам с методами:

− описание в классе всех необходимых методов базового задания;

− описание по крайней мере одной перегруженной операции;

***Разработка алгоритма***

*Задача 1*

Входные данные: *lives, px, py, ex, ey* – целые числа.

Выходные данные: *px, py, ex, ey* – целые числа.

*Player* – класс игрока, содержащий данные о нём и методы для управления им. Реализован следующим набором полей и методов:

*lives* – количество жизней

*x, y* – координаты игрока

*direction* – направление движения

*getX(), getY()*

Методы получения координат игрока.

Входные данные: отсутствуют

Выходные данные: целое число

*setDir()*

Метод установки направления движения.

Входные данные: два целых числа или объект класса *Direction*

Выходные данные: отсутствуют

*move()*

Метод передвижения игрока.

Входные данные: два целых числа

Выходные данные: отсутствуют

*Enemy* – класс игрока, содержащий данные о нём и методы для управления им. Реализован следующим набором полей и методов:

*x, y* – координаты врага

*direction* – направление движения

*getX(), getY()*

Методы получения координат.

Входные данные: отсутствуют

Выходные данные: целое число

*setDir()*

Метод установки направления движения.

Входные данные: два целых числа или объект класса *Direction*

Выходные данные: отсутствуют

*move()*

Метод передвижения врага.

Входные данные: два целых числа

Выходные данные: отсутствуют

*Direction* – класс вектор, показывающий направление. Реализован следующим набором полей:

*horizontal* – горизонтальная составляющая

*vertical* – вертикальная составляющая

Алгоритм решения задачи представлен на рисунке 1.

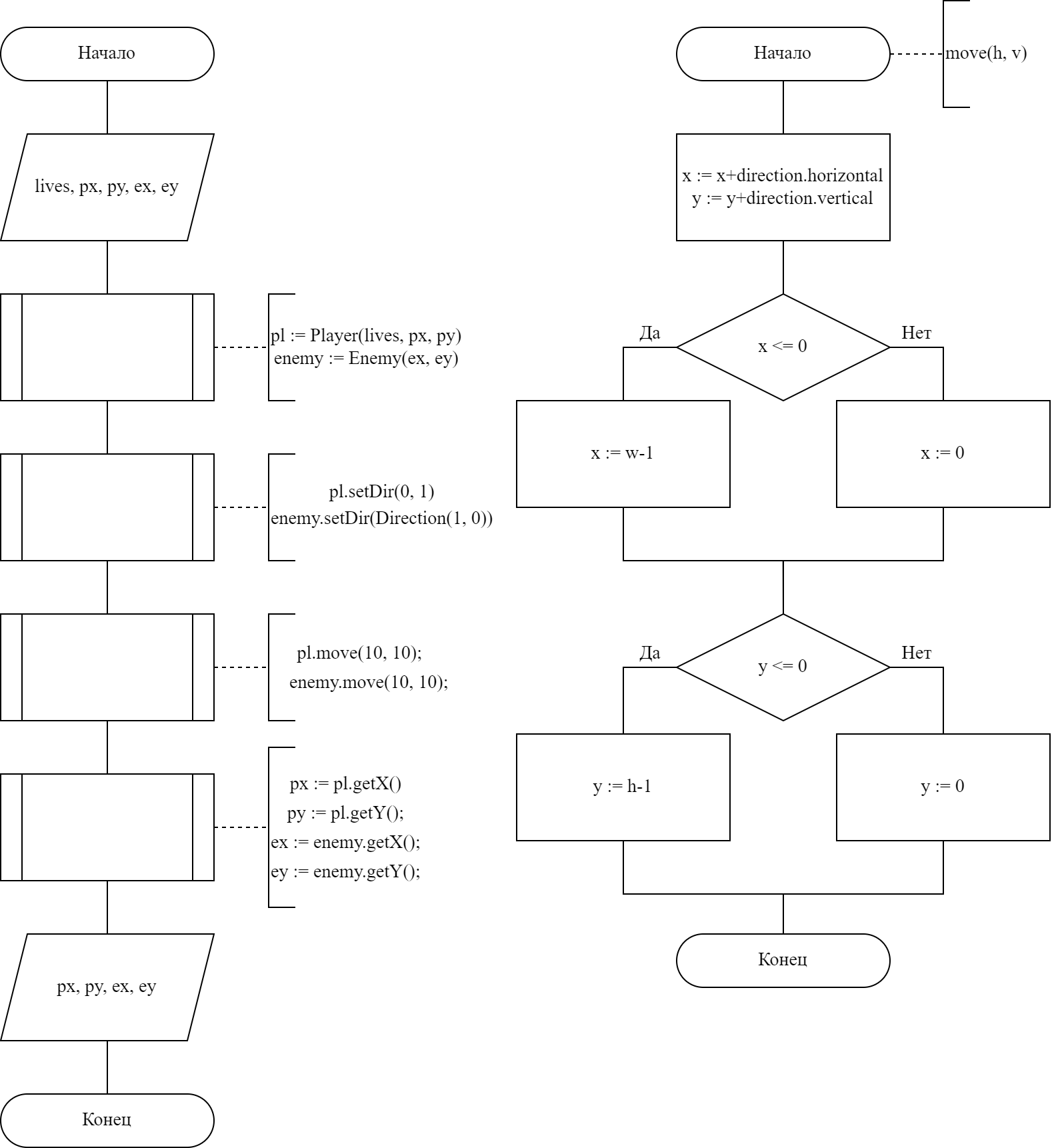


Рисунок 1 - Алгоритм решения задачи 1

***Текст программы***

*Текст программы для решения задачи 1*

*#ifndef DIRECTION\_H*

*#define DIRECTION\_H*

*class Direction {*

*public:*

*short horizontal;*

*short vertical;*

*Direction(int h, int v) {*

*this->horizontal = h;*

*this->vertical = v;*

*}*

*Direction() {*

*this->horizontal = 0;*

*this->vertical = 0;*

*}*

*};*

*#endif // DIRECTION\_H*

*#ifndef MOVABLE\_H*

*#define MOVABLE\_H*

*#include <direction.h>*

*#include <cstdlib>*

*class Movable*

*{*

*protected:*

*int x;*

*int y;*

*int memAnim;*

*Direction direction;*

*public:*

*float speed;*

*float movePhase;*

*void move(int h, int w) {*

*this->x += direction.horizontal;*

*this->y += direction.vertical;*

*if (this->x <= 0) {*

*this->x = w-1;*

*} else if (this->x >= w-1) {*

*this->x = 0;*

*}*

*if (this->y <= 0) {*

*this->y = h-1;*

*} else if (this->y >= h-1) {*

*this->y = 0;*

*}*

*};*

*int getX() {return x;};*

*int getY() {return y;};*

*void setDir(int h, int v) {*

*this->direction.horizontal = h;*

*this->direction.vertical = v;*

*};*

*void setDir(Direction dir) {*

*this->direction = dir;*

*};*

*int getH() {return this->direction.horizontal;};*

*int getV() {return this->direction.vertical;};*

*Direction getDir() {return this->direction;};*

*int getAnimDir() {*

*if (direction.horizontal == -1) {*

*memAnim = 2;*

*return 2;*

*}*

*else if (direction.horizontal == 1) {*

*memAnim = 0;*

*return 0;*

*}*

*else if (direction.vertical == -1) {*

*memAnim = 3;*

*return 3;*

*}*

*else if (direction.vertical == 1) {*

*memAnim = 1;*

*return 1;*

*}*

*else { return memAnim; };*

*}*

*Movable() {direction.horizontal = 0; direction.vertical = 0; memAnim = 0;};*

*Movable(int x, int y) {*

*this->x = x;*

*this->y = y;*

*direction.horizontal = 0;*

*direction.vertical = 0;*

*memAnim = 0;*

*};*

*};*

*class Player: public Movable {*

*private:*

*Direction memoryDirection;*

*int spawnX;*

*int spawnY;*

*public:*

*int lives;*

*bool targetable;*

*void setMDir(int h, int v) {*

*this->memoryDirection.horizontal = h;*

*this->memoryDirection.vertical = v;*

*};*

*void setMDir(Direction dir) {*

*this->memoryDirection = dir;*

*};*

*int getMH() {return this->memoryDirection.horizontal;};*

*int getMV() {return this->memoryDirection.vertical;};*

*Direction getMDir() {return this->memoryDirection;};*

*void toSpawn() {*

*targetable = false;*

*x = spawnX;*

*y = spawnY;*

*setDir(0, 0);*

*lives--;*

*movePhase = 0;*

*}*

*Player() {*

*direction.horizontal = 0;*

*direction.vertical = 0;*

*memoryDirection.horizontal = 0;*

*memoryDirection.vertical = 0;*

*x = 0;*

*y = 0;*

*movePhase = 0;*

*speed = 1.0;*

*lives = 3;*

*spawnX = x;*

*spawnY = y;*

*targetable = true;*

*}*

*Player(int lives): Player() {*

*this->lives = lives;*

*}*

*Player(int lives, int x, int y): Player(lives) {*

*this->x = x;*

*this->y = y;*

*spawnX = x;*

*spawnY = y;*

*}*

*Player(int lives, int x, int y, int h, int v): Player(lives, x, y) {*

*this->setDir(h, v);*

*}*

*Player(int lives, int x, int y, Direction dir): Player(lives, x, y) {*

*this->setDir(dir);*

*}*

*};*

*class Enemy: public Movable {*

*public:*

*int color;*

*Enemy() {*

*this->color = rand()%4;*

*this->movePhase = 0;*

*this->speed = 0.8;*

*direction.horizontal = 0;*

*direction.vertical = 0;*

*};*

*Enemy(int x, int y): Enemy() {*

*this->x = x;*

*this->y = y;*

*};*

*};*

*#endif // MOVABLE\_H*

*int main(){*

*int lives, px, py, ex, ey;*

*cin >> lives >> px >> py >> ex >> ey;*

*Player pl = Player(lives, px, py);*

*Enemy enemy = Enemy(ex, ey);*

*pl.setDir(0, 1);*

*enemy.setDir(Direction(1, 0));*

*pl.move(10, 10);*

*enemy.move(10, 10);*

*cout << "Player: " << pl.getX() << " " << pl.getY() << " Enemy: " << enemy.getX() << " " << enemy.getY() << endl;*

*}*

***Тестирование программы***

Тестирование задачи 1 представлено на рисунках 2-3.

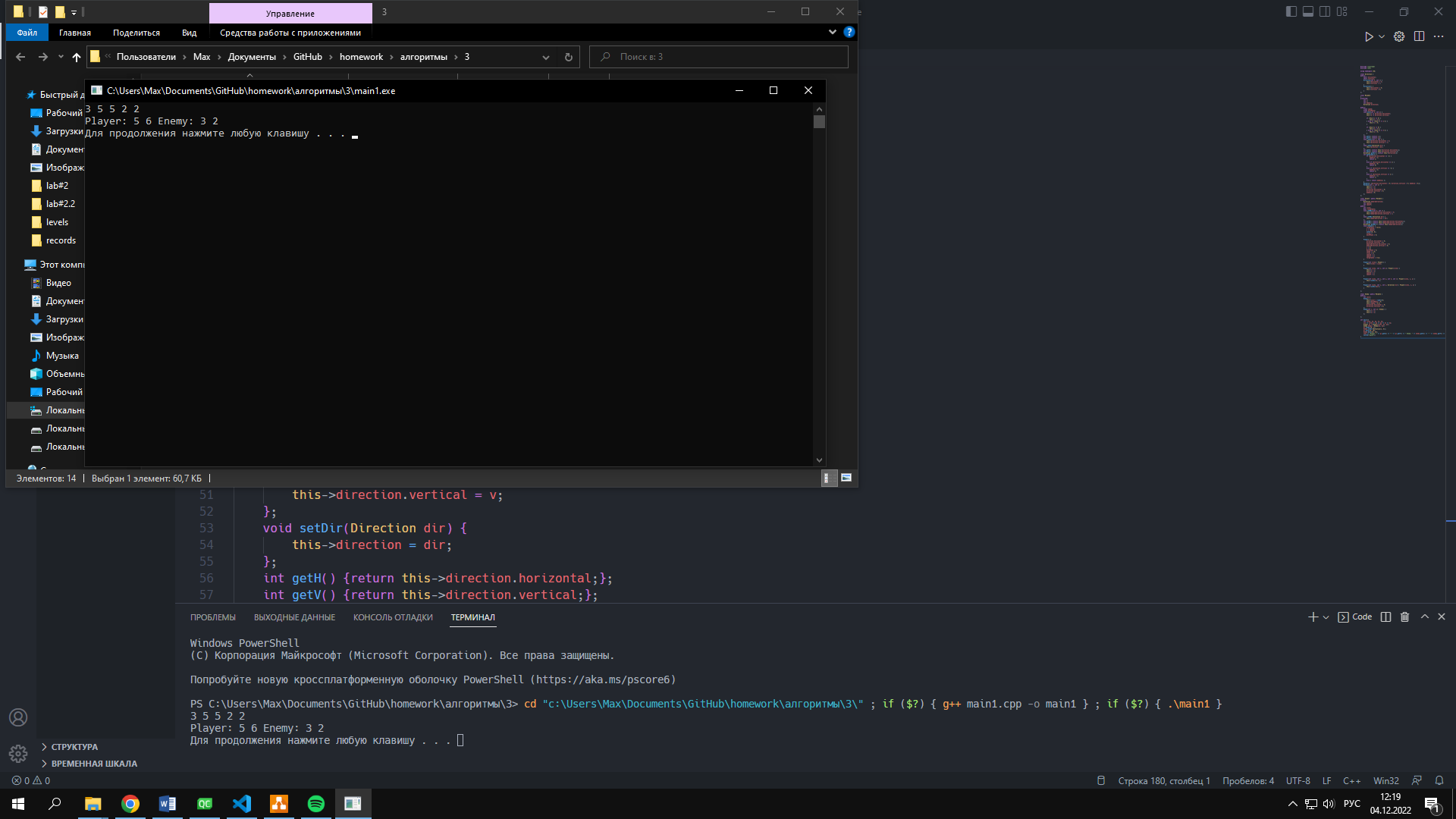


Рисунок 2 - Тест 1 задачи 1

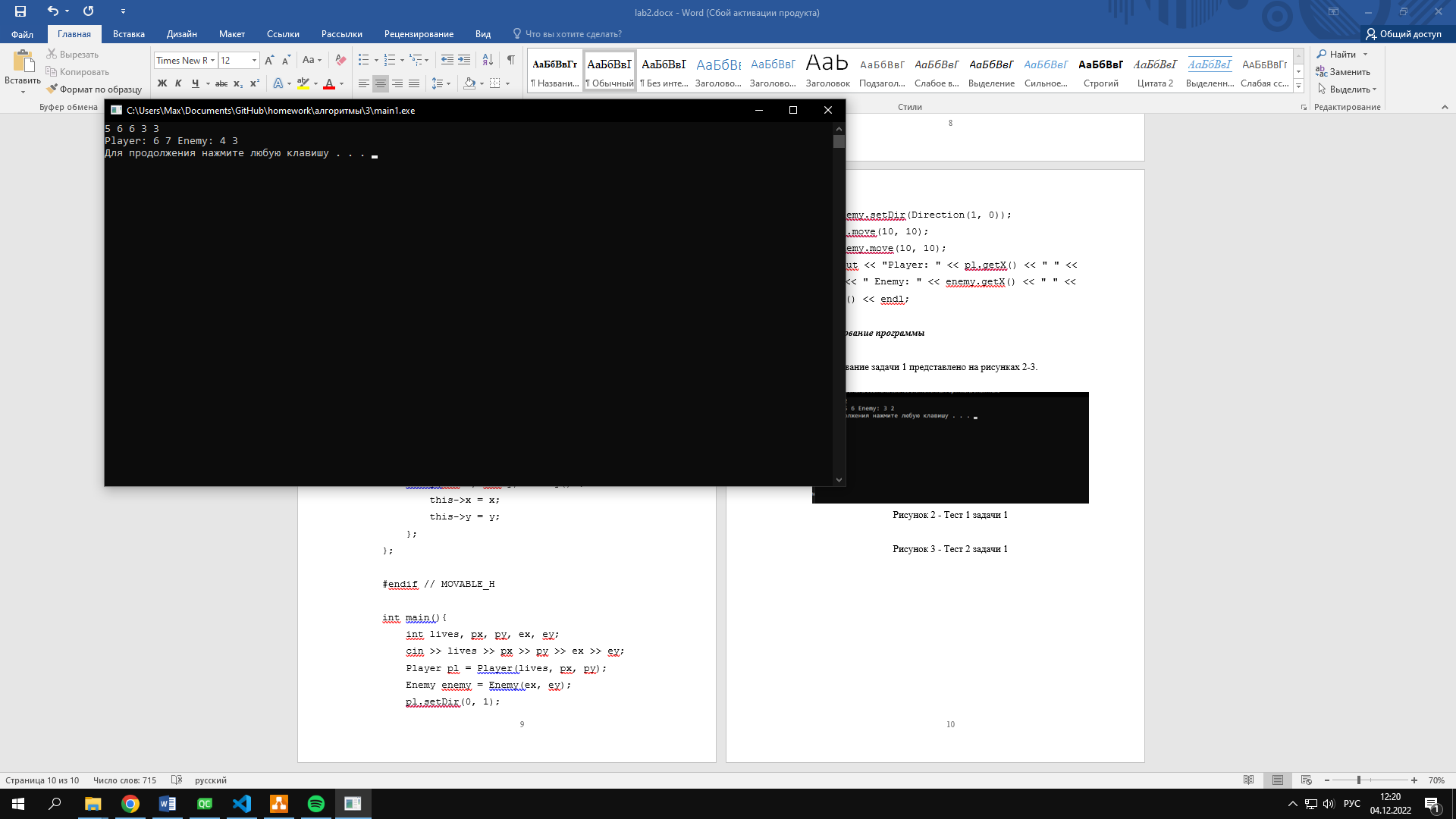


Рисунок 3 - Тест 2 задачи 1