ИСТ-АДМО-202Б

Трубин В.Д.

Орлов К.О.

3241.406115.000 ПЗ

Геоинформационных систем

Языки программирования

Уфа – 2025

Основы проектирования классов. Инкапсуляция.

Содержание

[Введение 3](#_Toc198797429)

[1. Задание 1 4](#_Toc198797430)

[2. Задание 2 7](#_Toc198797431)

[3. Задание 3 10](#_Toc198797432)

[Заключение 13](#_Toc198797433)

[Список литературы 14](#_Toc198797434)

Введение

Закрепление и углубление знаний, полученных в теоретической части дисциплины, развитие навыков объектно-ориентированной технологии проектирования алгоритмов и программ, развитие творческих способностей в области программирования.

# Задание 1

Создать класс для описания геометрических объектов в соответствии с вариантом. Для создаваемого класса предусмотреть четкое выполнение принципов Инкапсуляции (все данные объекта недоступны для прямого обращения). Все данные задаются и выводятся посредством вызова интерфейсных методов.

#include <iostream>

#include <cmath>

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

class Rhombus {

private:

// Координаты вершин

double ax, ay;

double bx, by;

double cx, cy;

double dx, dy;

// Геометрические параметры

double side = 0.0;

double shortDiagonal = 0.0;

double longDiagonal = 0.0;

double acuteAngle = 0.0; // в градусах

double obtuseAngle = 0.0; // в градусах

double perimeter = 0.0;

double area = 0.0;

// Пересчет значений

void calculateAnglesAndDiagonals() {

if (side > 0 && shortDiagonal > 0) {

// длинная диагональ по теореме Пифагора

longDiagonal = 2 \* sqrt(side \* side - (shortDiagonal / 2.0) \* (shortDiagonal / 2.0));

// острый угол

acuteAngle = 2 \* asin(shortDiagonal / (2 \* side)) \* 180.0 / M\_PI;

obtuseAngle = 180.0 - acuteAngle;

// площадь

area = (shortDiagonal \* longDiagonal) / 2.0;

// периметр

perimeter = 4 \* side;

}

}

public:

// Сеттеры

void setSide(double a) {

if (a > 0) {

side = a;

calculateAnglesAndDiagonals();

}

}

void setShortDiagonal(double d1) {

if (d1 > 0 && d1 < 2 \* side) {

shortDiagonal = d1;

calculateAnglesAndDiagonals();

}

}

void setCoordinates(double ax\_, double ay\_, double bx\_, double by\_,

double cx\_, double cy\_, double dx\_, double dy\_) {

ax = ax\_; ay = ay\_;

bx = bx\_; by = by\_;

cx = cx\_; cy = cy\_;

dx = dx\_; dy = dy\_;

}

// Геттеры

void printDiagonals() const {

std::cout << "Short diagonal: " << shortDiagonal << std::endl;

std::cout << "Long diagonal: " << longDiagonal << std::endl;

}

void printPerimeter() const {

std::cout << "Perimeter: " << perimeter << std::endl;

}

void printArea() const {

std::cout << "Area: " << area << std::endl;

}

};

int main() {

Rhombus rh;

rh.setSide(5);

rh.setShortDiagonal(6);

rh.printDiagonals();

rh.printPerimeter();

rh.printArea();

return 0;

}

# Задание 2

Для класса объекта, полученного в задании 1 разработать private методы, вычисляющие недостающие (не вводимые в сетторах) данные.

#include <iostream>

#include <cmath>

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

class Rhombus {

private:

// Координаты точек (не используются в расчётах, можно использовать позже)

double ax, ay, bx, by, cx, cy, dx, dy;

// Геометрические характеристики

double side = 0.0;

double shortDiagonal = 0.0;

double longDiagonal = 0.0;

double acuteAngle = 0.0;

double obtuseAngle = 0.0;

double perimeter = 0.0;

double area = 0.0;

// === PRIVATE METHODS ===

// Вычисление длинной диагонали по теореме Пифагора

void calculateLongDiagonal() {

if (side > 0 && shortDiagonal > 0 && shortDiagonal < 2 \* side) {

longDiagonal = 2 \* sqrt(side \* side - (shortDiagonal \* shortDiagonal) / 4);

}

}

// Вычисление периметра

void calculatePerimeter() {

if (side > 0) {

perimeter = 4 \* side;

}

}

// Вычисление площади по формуле: (d1 \* d2) / 2

void calculateArea() {

if (shortDiagonal > 0 && longDiagonal > 0) {

area = (shortDiagonal \* longDiagonal) / 2;

}

}

// Вычисление углов (в градусах)

void calculateAngles() {

if (side > 0 && shortDiagonal > 0 && shortDiagonal < 2 \* side) {

acuteAngle = 2 \* asin(shortDiagonal / (2 \* side)) \* 180.0 / M\_PI;

obtuseAngle = 180.0 - acuteAngle;

}

}

// Централизованный пересчёт

void updateGeometry() {

calculateLongDiagonal();

calculatePerimeter();

calculateArea();

calculateAngles();

}

public:

// === SETTERS ===

void setSide(double a) {

if (a > 0) {

side = a;

updateGeometry();

}

}

void setShortDiagonal(double d1) {

if (d1 > 0) {

shortDiagonal = d1;

updateGeometry();

}

}

void setCoordinates(double ax\_, double ay\_, double bx\_, double by\_,

double cx\_, double cy\_, double dx\_, double dy\_) {

ax = ax\_; ay = ay\_;

bx = bx\_; by = by\_;

cx = cx\_; cy = cy\_;

dx = dx\_; dy = dy\_;

}

// === GETTERS ===

void printDiagonals() const {

std::cout << "Short diagonal: " << shortDiagonal << std::endl;

std::cout << "Long diagonal: " << longDiagonal << std::endl;

}

void printPerimeter() const {

std::cout << "Perimeter: " << perimeter << std::endl;

}

void printArea() const {

std::cout << "Area: " << area << std::endl;

}

void printAngles() const {

std::cout << "Acute angle: " << acuteAngle << " degrees\n";

std::cout << "Obtuse angle: " << obtuseAngle << " degrees\n";

}

};

int main() {

Rhombus rh;

rh.setSide(5);

rh.setShortDiagonal(6);

rh.printDiagonals();

rh.printPerimeter();

rh.printArea();

rh.printAngles();

return 0;

}

# Задание 3

Разработать конструктор с параметром, принимающий значения величин согласно варианта задания и выполняющий вычисление всех недостающих параметров путем вызова соответствующих private-методов. Пересмотреть интерфейсные методы (сетторы), на предмет сохранения целостности данных при изменении устанавливаемого параметра.

#include <iostream>

#include <cmath>

#ifndef M\_PI

#define M\_PI 3.14159265358979323846

#endif

class Rhombus {

private:

// Геометрические характеристики

double side = 0.0;

double shortDiagonal = 0.0;

double longDiagonal = 0.0;

double perimeter = 0.0;

double area = 0.0;

double acuteAngle = 0.0;

double obtuseAngle = 0.0;

// === PRIVATE METHODS ===

void calculateLongDiagonal() {

longDiagonal = 2 \* sqrt(side \* side - (shortDiagonal \* shortDiagonal) / 4.0);

}

void calculatePerimeter() {

perimeter = 4 \* side;

}

void calculateArea() {

area = (shortDiagonal \* longDiagonal) / 2.0;

}

void calculateAngles() {

acuteAngle = 2 \* asin(shortDiagonal / (2 \* side)) \* 180.0 / M\_PI;

obtuseAngle = 180.0 - acuteAngle;

}

void updateGeometry() {

if (side > 0 && shortDiagonal > 0 && shortDiagonal < 2 \* side) {

calculateLongDiagonal();

calculatePerimeter();

calculateArea();

calculateAngles();

}

else {

longDiagonal = perimeter = area = acuteAngle = obtuseAngle = 0;

}

}

public:

// === PARAMETERIZED CONSTRUCTOR ===

Rhombus(double a, double d1) {

if (a > 0 && d1 > 0 && d1 < 2 \* a) {

side = a;

shortDiagonal = d1;

updateGeometry();

}

else {

std::cerr << "Invalid values for rhombus. Side and diagonal must be positive, and diagonal < 2 \* side.\n";

}

}

// === SETTERS (обновлённые, с защитой) ===

void setSide(double a) {

if (a > 0 && shortDiagonal < 2 \* a) {

side = a;

updateGeometry();

}

else {

std::cerr << "Invalid side value. Must be > 0 and satisfy diagonal < 2 \* side.\n";

}

}

void setShortDiagonal(double d1) {

if (d1 > 0 && d1 < 2 \* side) {

shortDiagonal = d1;

updateGeometry();

}

else {

std::cerr << "Invalid diagonal value. Must be > 0 and < 2 \* side.\n";

}

}

// === GETTERS ===

void printDiagonals() const {

std::cout << "Short diagonal: " << shortDiagonal << std::endl;

std::cout << "Long diagonal: " << longDiagonal << std::endl;

}

void printPerimeter() const {

std::cout << "Perimeter: " << perimeter << std::endl;

}

void printArea() const {

std::cout << "Area: " << area << std::endl;

}

void printAngles() const {

std::cout << "Acute angle: " << acuteAngle << " degrees\n";

std::cout << "Obtuse angle: " << obtuseAngle << " degrees\n";

}

};

int main() {

// Создание объекта с параметрами

Rhombus rh(5, 6);

rh.printDiagonals();

rh.printPerimeter();

rh.printArea();

rh.printAngles();

std::cout << "\nChange side to 6:\n";

rh.setSide(6); // Пересчёт всех значений

rh.printDiagonals();

rh.printArea();

return 0;

}

Заключение

Входе данной практической работы были изучены основные принципы объектно-ориентированного программирования. В результате работы была создана программа, демонстрирующая основы ООП на практике.

Список литературы

1. Кубенский, А. А. Функциональное программирование : учебник и практикум для академического бакалавриата / А. А. Кубенский. — М. : Издательство Юрайт, 2019. — 348 с.
2. Трофимов, В. В. Основы алгоритмизации и программирования : учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. — М. : Издательство Юрайт, 2019. — 137 с