Задача 14. Классы

1. Класс – комплексное число. Методы – сумма, разность, произведение комплексных чисел. Поля – действительная и мнимая часть комплексного числа.

#include <iostream>

class ComplexNumber {

private:

double real\_part;

double imaginary\_part;

public:

ComplexNumber(double real\_part = 0.0, double imaginary\_part = 0.0)

: real\_part(real\_part), imaginary\_part(imaginary\_part) {}

ComplexNumber operator+(const ComplexNumber& other) const {

return ComplexNumber(real\_part + other.real\_part, imaginary\_part + other.imaginary\_part);

}

ComplexNumber operator-(const ComplexNumber& other) const {

return ComplexNumber(real\_part - other.real\_part, imaginary\_part - other.imaginary\_part);

}

ComplexNumber operator\*(const ComplexNumber& other) const {

return ComplexNumber(real\_part \* other.real\_part - imaginary\_part \* other.imaginary\_part,

real\_part \* other.imaginary\_part + imaginary\_part \* other.real\_part);

}

void print() const {

std::cout << "(" << real\_part << " + " << imaginary\_part << "i)" << std::endl;

}

};

int main() {

ComplexNumber a(2.0, 3.0);

ComplexNumber b(4.0, -5.0);

ComplexNumber sum = a + b;

std::cout << "Сумма: ";

sum.print();

ComplexNumber difference = a - b;

std::cout << "Разность: ";

difference.print();

ComplexNumber product = a \* b;

std::cout << "Произведение: ";

product.print();

return 0;

}

/////////////////////////////////////////////////////////////////////////////////////////////////

2. Класс – десятичная дробь. Методы класса – сумма, разность, произведение, частное десятичных дробей. Поля класса – целая и дробная часть числа.

class DecimalFraction {

private:

int integer\_part;

double fractional\_part;

public:

DecimalFraction(int integer\_part = 0, double fractional\_part = 0.0)

: integer\_part(integer\_part), fractional\_part(fractional\_part) {}

DecimalFraction operator+(const DecimalFraction& other) const;

DecimalFraction operator-(const DecimalFraction& other) const;

DecimalFraction operator\*(const DecimalFraction& other) const;

DecimalFraction operator/(const DecimalFraction& other) const;

void print() const;

};

/////////////////////////////////////////////////////////////////////////////////////////////////

3. Класс – натуральная дробь. Методы класса – сумма, разность, произведение, частное натуральных дробей. Поля класса – числитель и знаменатель дроби.

class Fraction {

private:

int numerator;

int denominator;

public:

Fraction(int numerator = 0, int denominator = 1)

: numerator(numerator), denominator(denominator) {}

Fraction operator+(const Fraction& other) const;

Fraction operator-(const Fraction& other) const;

Fraction operator\*(const Fraction& other) const;

Fraction operator/(const Fraction& other) const;

};

/////////////////////////////////////////////////////////////////////////////////////////////////

4. Класс – строка символов. Методы класса – расчет длины строки и количества слов. Поля класса – строка, длина строки, кол–во слов в строке.

class String {

private:

std::string str;

int length;

int word\_count;

public:

String(const std::string& str = "")

: str(str), length(str.length()), word\_count(countWords(str)) {}

int getLength() const { return length; }

int getWordCount() const { return word\_count; }

private:

static int countWords(const std::string& str);

};

/////////////////////////////////////////////////////////////////////////////////////////////////

5. Класс – отрезок. Методы – расчет середины и длины. Поля класса – координаты на- чала, конца, середины и длина.

class Segment {

private:

double start;

double end;

double middle;

double length;

public:

Segment(double start = 0.0, double end = 0.0)

: start(start), end(end), middle((start + end) / 2), length(end - start) {}

double getMiddle() const { return middle; }

double getLength() const { return length; }

};

/////////////////////////////////////////////////////////////////////////////////////////////////

6. Класс – окружность. Методы – расчет длины окружности и ее площади. Поля класса – координаты центра, радиус, длина, площадь окружности.

class Circle {

private:

double x;

double y;

double radius;

double circumference;

double area;

public:

Circle(double x = 0.0, double y = 0.0, double radius = 0.0)

: x(x), y(y), radius(radius), circumference(2 \* M\_PI \* radius),

area(M\_PI \* radius \* radius) {}

double getCircumference() const { return circumference; }

double getArea() const { return area; }

};

/////////////////////////////////////////////////////////////////////////////////////////////////

7.Класс – квадратное уравнение. Методы – расчет корней уравнения. Поля – корни уравнения, коэффициенты *a* , *b* , *c* .

class QuadraticEquation {

private:

double root1;

double root2;

double a;

double b;

double c;

public:

QuadraticEquation(double a = 0.0, double b = 0.0, double c = 0.0)

: root1(0.0), root2(0.0), a(a), b(b), c(c) {

double discriminant = b \* b - 4 \* a \* c;

if (discriminant >= 0) {

root1 = (-b + sqrt(discriminant)) / (2 \* a);

root2 = (-b - sqrt(discriminant)) / (2 \* a);

}

}

double getRoot1() const { return root1; }

double getRoot2() const { return root2; }

};

/////////////////////////////////////////////////////////////////////////////////////////////////

8. Класс – шар. Методы – расчет площади и объема шара. Поля – координаты центра (*x*, *y*, *z*),радиусшара.

#include <cmath>

class Sphere {

private:

double x;

double y;

double z;

double radius;

public:

Sphere(double x = 0.0, double y = 0.0, double z = 0.0, double radius = 0.0)

: x(x), y(y), z(z), radius(radius) {}

double getSurfaceArea() const {

return 4 \* M\_PI \* radius \* radius;

}

double getVolume() const {

return 4.0 / 3.0 \* M\_PI \* radius \* radius \* radius;

}

};

/////////////////////////////////////////////////////////////////////////////////////////////////

9. Класс – конус. Методы – расчет площади и объема конуса. Поля – радиус основания, высота конуса.

#include <cmath>

class Cone {

private:

double radius;

double height;

public:

Cone(double radius = 0.0, double height = 0.0)

: radius(radius), height(height) {}

double getSurfaceArea() const {

double slant\_height = sqrt(radius \* radius + height \* height);

return M\_PI \* radius \* slant\_height + M\_PI \* radius \* radius;

}

double getVolume() const {

return 1.0 / 3.0 \* M\_PI \* radius \* radius \* height;

}

};

/////////////////////////////////////////////////////////////////////////////////////////////////

10. Класс – параллелепипед. Методы – расчет площади поверхности и объема параллеле- пипеда. Поля – ребра *a* , *b* , *c* , площадь и объем.

class Parallelepiped {

private:

double a, b, c;

public:

Parallelepiped(double a = 0.0, double b = 0.0, double c = 0.0)

: a(a), b(b), c(c) {}

double getSurfaceArea() const {

return 2 \* (a \* b + b \* c + c \* a);

}

double getVolume() const {

return a \* b \* c;

}

};

/////////////////////////////////////////////////////////////////////////////////////////////////

11. Класс – целочисленный вектор фиксированной размерности. Методы – сумма, раз- ность, скалярное произведение векторов. Поля – размерность вектора, координаты вектора.

#include <vector>

class IntVector {

private:

std::vector<int> coordinates;

public:

IntVector(const std::vector<int>& coords = {}) : coordinates(coords) {}

IntVector operator+(const IntVector& other) const;

IntVector operator-(const IntVector& other) const;

int dotProduct(const IntVector& other) const;

};

/////////////////////////////////////////////////////////////////////////////////////////////////

12. Класс – треугольник. Методы – расчет площади и периметра треугольника. Поля – стороны треугольника.

class Triangle {

private:

double side1;

double side2;

double side3;

public:

Triangle(double s1 = 0.0, double s2 = 0.0, double s3 = 0.0)

: side1(s1), side2(s2), side3(s3) {}

double getArea() const {

double s = (side1 + side2 + side3) / 2;

return sqrt(s \* (s - side1) \* (s - side2) \* (s - side3));

}

double getPerimeter() const {

return side1 + side2 + side3;

}

};

/////////////////////////////////////////////////////////////////////////////////////////////////

13. Класс – студент. Методы – определение среднего балла и решение о начислении сти- пендии. Поля – ФИО студента, оценки по дисциплинам в текущую сессию, средний балл.

#include <string>

#include <vector>

class Student {

private:

std::string fullName;

std::vector<int> grades;

double averageGrade;

public:

Student(const std::string& name = "", const std::vector<int>& grades = {})

: fullName(name), grades(grades), averageGrade(0.0) {

if (!grades.empty()) {

int sum = 0;

for (int grade : grades) {

sum += grade;

}

averageGrade = static\_cast<double>(sum) / grades.size();

}

}

double getAverageGrade() const { return averageGrade; }

bool isEligibleForScholarship() const {

return averageGrade >= 4.0; // Assuming a minimum required average grade of 4.0 for scholarship

}

};

/////////////////////////////////////////////////////////////////////////////////////////////////

14. Класс – трапеция. Методы – расчет площади и периметра трапеции. Поля – стороны трапеции, площадь и периметр.

class Trapezoid {

private:

double side1;

double side2;

double height;

public:

Trapezoid(double s1 = 0.0, double s2 = 0.0, double h = 0.0)

: side1(s1), side2(s2), height(h) {}

double getArea() const {

return (side1 + side2) \* height / 2.0;

}

double getPerimeter() const {

return side1 + side2 + 2 \* sqrt(height \* height + pow((side2 - side1) / 2, 2));

}

};

/////////////////////////////////////////////////////////////////////////////////////////////////

15. Класс – целочисленный вектор фиксированной размерности. Методы – сравнение двух векторов на равенство, вычисление длины вектора. Поля – размерность вектора, координаты вектора.

#include <vector>

class IntVector {

private:

std::vector<int> coordinates;

public:

IntVector(const std::vector<int>& coords = {}) : coordinates(coords) {}

bool operator==(const IntVector& other) const {

return coordinates == other.coordinates;

}

double getLength() const {

double sumOfSquares = 0.0;

for (int coordinate : coordinates) {

sumOfSquares += coordinate \* coordinate;

}

return sqrt(sumOfSquares);

}

};

/////////////////////////////////////////////////////////////////////////////////////////////////

Задача 15. Классы

ЗАДАНИЕ 1: для всех вариантов задач создать класс с указанными двумя полями (Поле 1, Поле 2) и тремя методами:

– конструктор для инициализации объекта;  
– функция формирования строки с информацией об объекте;  
– функция обработки значений полей по индивидуальному варианту.

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Вещественное число – левая граница диапазона | Вещественное число – правая граница диапазо- на | Квадрат длины диапазона |

#include <iostream>

#include <cmath>

#include <sstream>

class Range {

private:

double left, right;

public:

Range(double l, double r) : left(l), right(r) {}

std::string to\_string() const {

std::ostringstream oss;

oss << "[" << left << ", " << right << "]";

return oss.str();

}

double length\_squared() const {

double len = right - left;

return len \* len;

}

};

int main() {

Range r(1.5, 4.2);

std::cout << "Range: " << r.to\_string() << std::endl;

std::cout << "Length squared: " << r.length\_squared() << std::endl;

return 0;

}

/////////////////////////////////////////////////////////////////////////////////////////////////

ЗАДАНИЕ 2: для класса, созданного в предыдущем задании (по вариантам табл. 1) создать класс-потомок с дополнительным полем, указанным в индивидуальном задании (табл. 2, стол- бец 2).

Реализовать в классе-потомке методы:  
– конструктор;  
– функцию обработки данных, указанную в индивидуальном задании (табл. 2, столбец 3). Демонстрация работы включает ввод и вывод информации об объектах: классе-родителе

и классе-потомке.

5. Вещественное число *х* Проверить, принадлежит ли число *х* заданному диапазону

#include <iostream>

#include <cmath>

// Родительский класс Range

class Range {

protected:

double left; // левая граница диапазона

double right; // правая граница диапазона

public:

// Конструктор

Range(double l, double r) : left(l), right(r) {}

// Функция формирования строки с информацией об объекте

virtual std::string toString() {

return "Range [" + std::to\_string(left) + ", " + std::to\_string(right) + "]";

}

// Функция обработки значений полей по индивидуальному варианту

virtual double process() {

return std::pow(right - left, 2);

}

};

// Дочерний класс RangeWithCheck

class RangeWithCheck : public Range {

private:

double x; // проверяемое число

public:

// Конструктор

RangeWithCheck(double l, double r, double \_x) : Range(l, r), x(\_x) {}

// Функция формирования строки с информацией об объекте

std::string toString() override {

return "RangeWithCheck [" + std::to\_string(left) + ", " + std::to\_string(right) + "] with x = " + std::to\_string(x);

}

// Функция обработки значений полей по индивидуальному варианту

double process() override {

if (x >= left && x <= right) {

return 1;

} else {

return 0;

}

}

};

int main() {

// Создание объекта родительского класса

Range r(1.5, 4.2);

std::cout << r.toString() << " has length square of " << r.process() << std::endl;

// Создание объекта дочернего класса

RangeWithCheck rc(0.7, 3.9, 2.5);

std::cout << rc.toString() << " has x " << rc.process() << " in the range" << std::endl;

return 0;

}