Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.04 – «Программная инженерия»

Дисциплина: «Проектирование архитектуры программных систем»

Профиль: «Разработка программно-информационных систем»

Семестр 5

ОТЧЕТ

по лабораторной работе №2

Тема: «Паттерны проектирования»

Выполнил: студент группы РИС-22-1б

Поважный В.Е. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Викентьева О.Л. \_\_\_\_\_\_\_\_\_\_\_

Пермь, 2024

**Постановка задачи**

1. **Описание используемого паттерна**

**1.1 Название паттерна**

Паттерн "Шаблонный метод" (Template method)

**1.2 Контекст и условия применения**

**Шаблонный метод** — это поведенческий паттерн, который определяет основу алгоритма в методе базового класса и позволяет подклассам переопределять отдельные шаги алгоритма, не изменяя его структуру.

**Основные свойства:**

1. Базовый класс содержит шаблонный метод, определяющий последовательность выполнения алгоритма.
2. Шаги алгоритма, которые могут отличаться, делаются виртуальными и переопределяются в подклассах.
3. Это позволяет избежать дублирования кода и упростить добавление новых вариаций алгоритма.

**1.3 Решение**

В данной задаче паттерн "Шаблонный метод" поможет выделить общий алгоритм для итерационных методов нахождения корня уравнения f(x)=0f(x) = 0f(x)=0, а шаги вычисления Xk+1X\_{k+1}Xk+1​ будут разными для метода хорд и метода Ньютона.

Общий алгоритм:

1. Проверить, достигнута ли точность решения.
2. Вычислить следующий Xk+1X\_{k+1}Xk+1​ на основе текущего XkX\_kXk​.
3. Повторить до выполнения условия.
4. **Диаграмма классов**

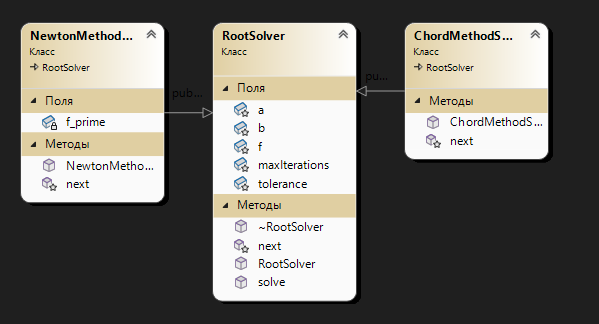


Рисунок 1 – Диаграмма классов

**Листинг**

#ifndef ROOTSOLVER\_H

#define ROOTSOLVER\_H

#include <iostream>

#include <cmath>

#include <functional>

#include <iostream>

#include <cmath>

#include <functional>

class RootSolver

{

protected:

double a, b;

double tolerance;

int maxIterations;

std::function<double(double)> f;

virtual double next(double x) = 0;

public:

RootSolver(double a, double b, double tol, int maxIter, std::function<double(double)> f)

: a(a), b(b), tolerance(tol), maxIterations(maxIter), f(f) {}

double solve(double initialGuess)

{

double x = initialGuess;

for (int i = 0; i < maxIterations; ++i)

{

if (std::abs(f(x)) < tolerance)

{

std::cout << "Converged in " << i + 1 << " iterations.\n";

return x;

}

std::cout << "Iteration " << i + 1 << ": x = " << x << ", f(x) = " << f(x) << "\n";

x = next(x);

}

throw std::runtime\_error("Exceeded maximum number of iterations without convergence.");

}

virtual ~RootSolver() = default;

};

#endif

#ifndef CHORDMETHODSOLVER\_H

#define CHORDMETHODSOLVER\_H

#include "RootSolver.h"

class ChordMethodSolver : public RootSolver

{

public:

ChordMethodSolver(double a, double b, double tol, int maxIter, std::function<double(double)> f)

: RootSolver(a, b, tol, maxIter, f) {}

protected:

double next(double x) override

{

double fa = f(a);

double fb = f(b);

if (fb == fa)

throw std::runtime\_error("Division by zero in Chord Method.");

return x - f(x) \* (b - a) / (fb - fa);

}

};

#endif

#ifndef NEWTONMETHODSOLVER\_H

#define NEWTONMETHODSOLVER\_H

#include "RootSolver.h"

class NewtonMethodSolver : public RootSolver

{

private:

std::function<double(double)> f\_prime;

public:

NewtonMethodSolver(double a, double b, double tol, int maxIter, std::function<double(double)> f, std::function<double(double)> f\_prime)

: RootSolver(a, b, tol, maxIter, f), f\_prime(f\_prime) {}

protected:

double next(double x) override

{

double derivative = f\_prime(x);

if (derivative == 0)

throw std::runtime\_error("Derivative is zero in Newton's Method.");

return x - f(x) / derivative;

}

};

#endif

#include <iostream>

#include <cmath>

#include "RootSolver.h"

#include "ChordMethodSolver.h"

#include "NewtonMethodSolver.h"

int main()

{

setlocale(LC\_ALL, "rus");

auto f = [](double x) { return x \* x - 2; };

auto f\_prime = [](double x) { return 2 \* x; };

double a = 0, b = 2, tolerance = 1e-6;

int maxIterations = 100;

try

{

ChordMethodSolver chordSolver(a, b, tolerance, maxIterations, f);

double chordRoot = chordSolver.solve(b);

std::cout << "Метод хорд: корень = " << chordRoot << std::endl;

NewtonMethodSolver newtonSolver(a, b, tolerance, maxIterations, f, f\_prime);

double newtonRoot = newtonSolver.solve(b);

std::cout << "Метод Ньютона: корень = " << newtonRoot << std::endl;

}

catch (const std::exception& e)

{

std::cerr << "Ошибка: " << e.what() << std::endl;

}

return 0;

}