МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Лабораторная работа №7
по дисциплине: Объектно-ориентированное программирование
Тема: Исключительные ситуации в C++.

Выполнил: студент группы ПВ-223
Дмитриев А.А.
Проверил:
Черников С.В.

Белгород 2024 г.

Цель работы: Получение теоретических знаний об исключительных ситуациях в C++. Получение практических навыков при работе с исключениями в C++.

Задание:

1. Изучить теоретические сведения об исключениях в C++.
2. Изучить самостоятельно стандартные классы для исключений предусмотренных в C++.
3. Разработать программу в соответствии с заданным вариантом задания.
4. Оформить отчет.

solve.h

```
#pragma once

#include <iostream>
#include <sstream>
#include <cmath>
#include <vector>
#include <exception>

class EquationSolver {
public:
    EquationSolver() {}
    virtual void solveEquation() = 0;
    void startInput() {
        while (true) {
            std::cout << "Введите параметры.\n";
            std::string args;
            std::getline(std::cin, args);
            std::stringstream ss(args);

            std::string arg;
            while (ss >> arg)
                params.push_back(stod(arg));

            try {
                solveEquation();
            }
            catch (NotSolveException& e) {
                std::cerr << "NotSolveException " << e.what();
                break;
            }
            catch (BadInputException& e) {
                std::cerr << "BadInputException " << e.what();
                break;
            }
        }
    }

    class NotSolveException : public std::exception
    {
    public:
        NotSolveException(const char* msg) : std::exception(msg) {}
    };

    class BadInputException : public std::exception
    {
    public:
        BadInputException(const char* msg) : std::exception(msg) {}
    };
```

```
protected:
    std::vector<double> params;
};

class QuadraticEquationSolver : public EquationSolver {
public:
    QuadraticEquationSolver() {}

    QuadraticEquationSolver(double a, double b, double c) {
        params.clear();
        params.push_back(a);
        params.push_back(b);
        params.push_back(c);
    }

    void solveEquation() override {
        double a, b, c;
        if (params.size() == 3) {
            a = params[0];
            b = params[1];
            c = params[2];
        }
        else {
            throw BadInputException("Превышено количество аргументов\n");
        }

        if (a == 0) throw BadInputException("a = 0");

        double discriminant = b * b - 4 * a * c;
        if (discriminant > 0) {
            double x1 = (-b + sqrt(discriminant)) / (2 * a);
            double x2 = (-b - sqrt(discriminant)) / (2 * a);
            std::cout << "Уравнение имеет два корня: x1 = " << x1 << ", x2 = " << x2 <<
std::endl;
        }
        else if (discriminant == 0) {
            double x = -b / (2 * a);
            std::cout << "Уравнение имеет один корень: x = " << x << std::endl;
        }
        else {
            throw NotSolveException("Уравнение не имеет действительных корней\n");
        }
    }
};
```

main.cpp

```
#include "solve.h"
int main() {
    setlocale(LC_ALL, "Russian");

    QuadraticEquationSolver solver;
    solver.startInput();

    QuadraticEquationSolver solver1(1, -3, 2);
    solver1.solveEquation();

    try {
        QuadraticEquationSolver solver2(0, 2, 3);
        solver2.solveEquation();
    }
    catch (...) {
        std::cerr << "error\n";
    }

    try {
    QuadraticEquationSolver solver3(1, 0, 2);
    solver3.solveEquation();
    }
```

```
    catch (...) {
        std::cerr << "error\n";
    }

    QuadraticEquationSolver solver4(1, -3, 0);
    solver4.solveEquation();

    QuadraticEquationSolver solver5(1, 0, 0);
    solver5.solveEquation();

    try {
        QuadraticEquationSolver solver6(1, 2, 3);
        solver6.solveEquation();
    }
    catch (...) {
        std::cerr << "error\n";
    }

    return 0;
}
```

Пример работы

```
Введите параметры.
1 4 1
Уравнение имеет два корня: x1 = -0.267949, x2 = -3.73205
Введите параметры.
5 5 5
BadInputException Превышено количество аргументов
Уравнение имеет два корня: x1 = 2, x2 = 1
error
error
Уравнение имеет два корня: x1 = 3, x2 = 0
Уравнение имеет один корень: x = -0
error
```

Вывод: В ходе лабораторной работы получили теоретические знания об исключительных ситуациях в C++. Получили практические навыки при работе с исключениями в C++.