**Отчёт по лабораторной работе №4**

**Основные конструкции языка Python.**

**Цель лабораторной работы: изучение основных конструкций языка Python.**

**Задание:**

Разработать программу для решения биквадратного уравнения.

Программа должна быть разработана в виде консольного приложения на языке Python.

Программа осуществляет ввод с клавиатуры коэффициентов А, В, С, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).

Коэффициенты А, В, С могут быть заданы в виде параметров командной строки ( вариант задания параметров приведен в конце файла с примером кода ). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. Описание работы с параметрами командной строки.

Если коэффициент А, В, С введен или задан в командной строке некорректно, то необходимо проигнорировать некорректное значение и вводить коэффициент повторно пока коэффициент не будет введен корректно. Корректно заданный коэффициент - это коэффициент, значение которого может быть без ошибок преобразовано в действительное число.

Разработайте программу на любом другом языке программирования (кроме С++). Выбран был язык программирования Rust.

**Текст программы**

use std::io;

fn get\_coef(prompt: &str) -> f64 {

    let mut input = String::new();

    println!("{}", prompt);

    io::stdin().read\_line(&mut input).expect("A string");

    return input.trim().parse::<f64>().expect("Введено не число!");

}

fn get\_roots(a: f64, b: f64, c: f64) -> Vec<f64>{

    let mut roots\_list = Vec::<f64>::new();

    let d: f64 = b\*b - 4.0\*a\*c;

    if d == 0.0 {

        if -b/(2.0\*a) > 0.0 {

            let root1: f64 = f64::sqrt(-b/(2.0\*a) as f64);

            let root2: f64 = -root1;

            roots\_list.push(root1);

            roots\_list.push(root2);

        }

        else if (a != 0.0) && (b == 0.0) {

            let root1: f64 = 0.0;

            roots\_list.push(root1)

        }

    }

    if d > 0.0 {

        let buf1: f64 = (-b - f64::sqrt(d as f64))/(2.0\*a);

        let buf2: f64 = (-b + f64::sqrt(d as f64))/(2.0\*a);

        if buf1 > 0.0 {

            let root1: f64 = f64::sqrt(buf1 as f64);

            let root2: f64 = -root1;

            roots\_list.push(root1);

            roots\_list.push(root2);

        }

        else if buf1 == 0.0 {

            let root1: f64 = 0.0;

            roots\_list.push(root1);

        }

        if buf2 > 0.0 {

            let root3: f64 = f64::sqrt(buf2 as f64);

            let root4: f64 = -root3;

            roots\_list.push(root3);

            roots\_list.push(root4);

        }

        else if buf2 == 0.0 {

            let root3: f64 = 0.0;

            roots\_list.push(root3);

        }

    }

    return roots\_list;

}

fn main() {

    let a = get\_coef("Введите коэффициент а: ");

    let b = get\_coef("Введите коэффициент b: ");

    let c = get\_coef("Введите коэффициент c: ");

    let roots = get\_roots(a, b, c);

    let len\_roots = roots.len();

    if len\_roots == 0 {

        println!("Корней нет");

    } else if len\_roots == 1 {

        println!("Один корень: {}", roots[0]);

    } else if len\_roots == 2 {

        println!("Два корня: {} и {}", roots[0], roots[1]);

    } else if len\_roots == 3 {

        println!("Три корня: {} и {} и {}", roots[0], roots[1], roots[2]);

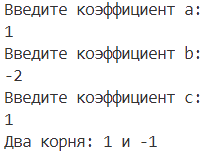
    } else {

        println!("Четыре корня: {} и {} и {} и {}", roots[0], roots[1], roots[2], roots[3]);

    }

}

**Пример выполнения программы**

****