**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Радиотехника»

Кафедра «Информатика и вычислительная техника»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1

«Основные конструкции языка Python»

Вариант «Реализация на Rust»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы РТ5-31Б: |  | преподаватель каф. ИУ5 |
| Щербинин А.О. |  | Гапанюк Ю.Е. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2023 г.

**ОПИСАНИЕ ЗАДАНИЯ.**

Разработать программу для решения [биквадратного уравнения.](https://ru.wikipedia.org/wiki/%D0%A3%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5_%D1%87%D0%B5%D1%82%D0%B2%D1%91%D1%80%D1%82%D0%BE%D0%B9_%D1%81%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D0%B8#%D0%91%D0%B8%D0%BA%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5)

1. Программа должна быть разработана в виде консольного приложения на языке Python.
2. Программа осуществляет ввод с клавиатуры коэффициентов А, В, С, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты А, В, С могут быть заданы в виде параметров командной строки ( [вариант задания параметров приведен в конце файла с примером кода](https://github.com/ugapanyuk/BKIT_2022/blob/main/code/lab1_code) ). Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2. [Описание работы с параметрами командной строки.](https://realpython.com/python-command-line-arguments/#the-command-line-interface)
4. 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.

**Текст программы**

use std::io;

use std::env;

type Double = f64;

enum Roots {

NoRoots,

OneRoot (Double),

TwoRoots (Double, Double),

ThreeRoots (Double, Double, Double),

FourRoots (Double, Double, Double, Double),

}

fn show\_roots(r: Roots){

match r{

Roots::NoRoots => println!("Нет корней"),

Roots::OneRoot(x) => println!("Один корень {}", x),

Roots::TwoRoots(x1, x2) => println!("Два корня: {} и {}", x1, x2),

Roots::ThreeRoots(x1, x2, x3) => println!("Три корня: {}, {} и {}", x1, x2, x3),

Roots::FourRoots(x1, x2, x3, x4) => println!("Четыре корня: {}, {}, {}, {}", x1, x2, x3, x4)

}

}

fn solve(tuple: (Double, Double, Double)) -> Roots{

let a = tuple.0;

let b = tuple.1;

let c = tuple.2;

let dis = b \* b - 4.0 \* a \* c;

if dis < 0.0 || dis == 0.0 && (-b)/(2.0 \* a) < 0.0 || a == 0.0{

Roots::NoRoots

}

else if dis == 0.0 && (-b)/(2.0 \* a) == 0.0{

Roots::OneRoot (0.0)

}

else if dis == 0.0{

let x = (-b)/(2.0 \* a);

Roots::TwoRoots(x.sqrt(), -x.sqrt())

}

else {

let x1 = (-b + dis.sqrt())/(2.0 \* a);

let x2 = (-b - dis.sqrt())/(2.0 \* a);

if x1 > 0.0 && x2 < 0.0{

Roots::TwoRoots(x1.sqrt(), -x1.sqrt())

}

else if x1 < 0.0 && x2 > 0.0{

Roots::TwoRoots(x2.sqrt(), -x2.sqrt())

}

else if x1 == 0.0 && x2 > 0.0{

Roots::ThreeRoots(0.0, x2.sqrt(), -x2.sqrt())

}

else if x1 > 0.0 && x2 == 0.0{

Roots::ThreeRoots(0.0, x1.sqrt(), -x1.sqrt())

}

else if x1 > 0.0 && x2 > 0.0{

Roots::FourRoots(x1.sqrt(), -x1.sqrt(), x2.sqrt(), -x2.sqrt())

}

else {

Roots::NoRoots

}

}

}

fn get\_3c() -> (Double, Double, Double){

let args: Vec<String> = env::args().collect();

if args.len() == 4{

let a = args[1].trim().parse().expect("Ошибка парсинга");

let b = args[2].trim().parse().expect("Ошибка парсинга");

let c = args[3].trim().parse().expect("Ошибка парсинга");

(a, b, c)

}

else {

println!("Введите коэффициент А:\n");

let a = get\_c();

println!("Введите коэффициент В:\n");

let b = get\_c();

println!("Введите коэффициент С:\n");

let c = get\_c();

(a, b, c)

}

}

fn get\_c() -> Double {

let mut number = String::new();

io::stdin().read\_line(&mut number).expect("Ввод некорректный!");

if let Ok(x) = number.trim().parse::<Double>() {

x

} else {

println!("Ошиба! Постарайтесь ввести все-же число:/");

get\_c()

}

}

fn main(){

show\_roots(solve(get\_3c()));

}

**Примеры выводов**







