Московский Государственный Технический
Университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчёт по лабораторной работе №3
«Основные конструкции языка Python (исполнение на Rust)»

Выполнил:
студент группы ИУ5-25б
Бикматов Д. А.

Проверил:
преподаватель
Гапанюк Ю. В.

Москва, 2024 г.

Постановка задачи:

Выполнить следующее задание на ЯП Rust.

Разработать программу для решения биквадратного уравнения.

1. Программа должна быть разработана в виде консольного приложения на языке Python.
2. Программа осуществляет ввод с клавиатуры коэффициентов A, B, C, вычисляет дискриминант и ДЕЙСТВИТЕЛЬНЫЕ корни уравнения (в зависимости от дискриминанта).
3. Коэффициенты A, B, C могут быть заданы в виде параметров командной строки. Если они не заданы, то вводятся с клавиатуры в соответствии с пунктом 2.
4. Если коэффициент A, B, C 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.
5. Дополнительное задание 1 (*). Разработайте две программы на языке Python - одну с применением процедурной парадигмы, а другую с применением объектно-ориентированной парадигмы.
6. Дополнительное задание 2 (*). Разработайте две программы - одну на языке Python, а другую на любом другом языке программирования (кроме C++).

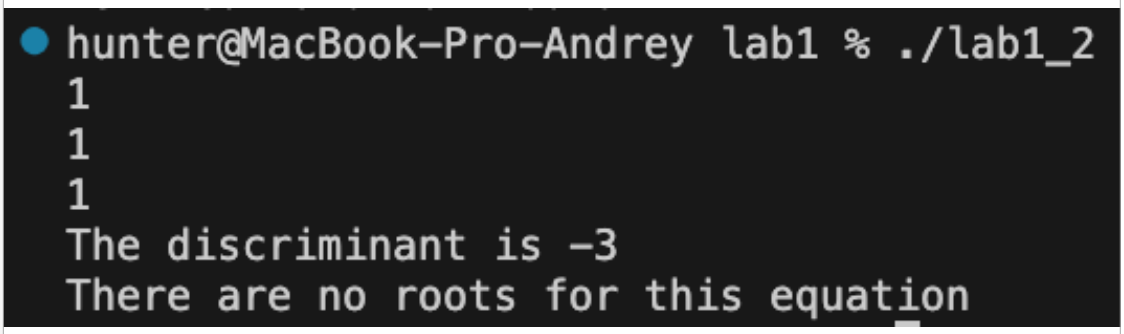
Текст программы:

```
use std::collections::LinkedList;

fn enter() -> f64 {
    loop {
        let mut input = String::new();
        std::io::stdin().read_line(&mut input).expect("Не удалось прочитать строку");
        match input.trim().parse::<f64>() {
            Ok(value) => return value,
            Err(_) => println!("Incorrect input"),
        }
    }
}

fn ans(a: f64, b: f64, dis: f64) -> LinkedList<f64> {
    let mut l: LinkedList<f64> = LinkedList::new();
    if dis == 0.0 && -b/(2.0*a) > 0.0{
        l.push_back(-(-b/(2.0*a)).sqrt());
```

```
        l.push_back((-b/(2.0*a)).sqrt());
    }
    if dis > 0.0{
        if ((-b-dis.sqrt())/(2.0*a)) > 0.0{
            l.push_back(-((-b-dis.sqrt())/(2.0*a)).sqrt());
            l.push_back(((-b-dis.sqrt())/(2.0*a)).sqrt());
        }
        if ((-b+dis.sqrt())/(2.0*a)) > 0.0{
            l.push_back(-((-b+dis.sqrt())/(2.0*a)).sqrt());
            l.push_back(((-b+dis.sqrt())/(2.0*a)).sqrt());
        }
    }
    return l;
}

fn main() {
    let args: Vec<String> = std::env::args().skip(1).collect();
    let mut args_f64: Vec<f64> = Vec::new();
    let mut flag = false;
    let a: f64;
    let b: f64;
    let c: f64;
    if args.len() == 3{
        flag = true;
        for i in 0..3{
            match args[i].trim().parse::<f64>() {
                Ok(value) => args_f64.push(value),
                Err(_) => flag = false,
            }
        }
    }
    if flag{
        a = args_f64[0];
        b = args_f64[1];
        c = args_f64[2];
    } else {
        a = enter();
        b = enter();
        c = enter();
    }
    let dis = b.powi(2) - 4.0*a*c;
    let res = ans(a, b, dis);
    println!("The discriminant is {}", dis);
    if res.is_empty() {
        println!("There are no roots for this equation");
```

```
    } else {
        println!("The roots are:");
        for i in res{
            println!("{}", i);
        }
    }
}
```

Пример выполнения:

Входные данные	Выходные данные
1 1 1	
1 -2 1