

**Московский государственный технический  
университет им. Н.Э. Баумана**

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

Курс «ПиКЯП»

Отчет по лабораторной работе №3

«Разработка на языке программирования Rust»

Выполнил:

студент группы ИУ5-36Б

Рухлин Алексей

Проверил:

преподаватель каф.  
ИУ5

Нардид А. Н.

Москва, 2024 г.

## Описание задания

### Задание:

Реализуйте любое из заданий курса на языке программирования Rust.

## Описание задания

Разработать программу для решения квадратного уравнения вида  $ax^2+bx+c=0$  с использованием языка программирования Rust. Программа должна:

1. Принимать коэффициенты  $a$ ,  $b$ ,  $c$  как параметры командной строки. Если параметры не указаны, программа запрашивает их ввод у пользователя.
2. Вычислять дискриминант и определять количество действительных корней уравнения.
3. В случае отсутствия корней выводить соответствующее сообщение.
4. Обрабатывать некорректный ввод, повторно запрашивая значения у пользователя.

---

## Текст программы

```
Rust
use std::env;
use std::io;
use std::process::exit;
```

```

fn get_coefficient_input(prompt: &str) -> f64 {
    loop {
        println!("{}", prompt);
        let mut input = String::new();
        io::stdin()
            .read_line(&mut input)
            .expect("Не удалось прочитать ввод");

        match input.trim().parse::<f64>() {
            Ok(value) => return value,
            Err(_) => println!("Некорректное значение.
Пожалуйста, введите число."),
        }
    }
}

```

```

fn get_coefficient_from_args_or_input(args: &[String],
index: usize, prompt: &str) -> f64 {
    if args.len() > index {
        match args[index].parse::<f64>() {
            Ok(value) => value,
            Err(_) => {
                println!("Некорректное значение
параметра {}. Пожалуйста, введите вручную.", prompt);
                get_coefficient_input(prompt)
            }
        }
    } else {
        get_coefficient_input(prompt)
    }
}

```

```
}
```

```
fn calculate_discriminant(a: f64, b: f64, c: f64) ->
f64 {
    b * b - 4.0 * a * c
}
```

```
fn main() {
    let args: Vec<String> = env::args().collect();

    let a = get_coefficient_from_args_or_input(&args,
1, "Введите коэффициент A:");
    if a == 0.0 {
        println!("Коэффициент A не может быть равен
нулю, это не квадратное уравнение.");
        exit(1);
    }

    let b = get_coefficient_from_args_or_input(&args,
2, "Введите коэффициент B:");
    let c = get_coefficient_from_args_or_input(&args,
3, "Введите коэффициент C:");

    let discriminant = calculate_discriminant(a, b, c);
    println!("Дискриминант: {}", discriminant);

    if discriminant > 0.0 {
        let x1 = (-b + discriminant.sqrt()) / (2.0 *
a);
```

```
        let x2 = (-b - discriminant.sqrt()) / (2.0 *  
a);  
        println!("Два действительных корня: x1 = {}, x2  
= {}", x1, x2);  
    } else if discriminant == 0.0 {  
        let x = -b / (2.0 * a);  
        println!("Один действительный корень: x = {}",  
x);  
    } else {  
        println!("Действительных корней нет.");  
    }  
}
```

---

## Экранные формы с примерами выполнения программы

### Пример 1 (Два корня):

Введите коэффициент A: 1

Введите коэффициент B: -3

Введите коэффициент C: 2

Дискриминант: 1.0

Два действительных корня: x1 = 2.0, x2 = 1.0

1.

### Пример 2 (Один корень):

Введите коэффициент A: 1

Введите коэффициент B: -2

Введите коэффициент C: 1

Дискриминант: 0.0

Один действительный корень:  $x = 1.0$

### Пример 3 (Нет корней):

Введите коэффициент A: 1

Введите коэффициент B: 0

Введите коэффициент C: 1

Дискриминант:  $-4.0$

Действительных корней нет.

2.

### Пример 4 (Командная строка):

\$ cargo run 1 -3 2

Дискриминант:  $1.0$

Два действительных корня:  $x_1 = 2.0$ ,  $x_2 = 1.0$

### Вывод

В ходе выполнения лабораторной работы была разработана программа на языке Rust для решения квадратных уравнений.

В процессе разработки были выполнены следующие задачи:

- Реализован ввод коэффициентов через командную строку или консоль с проверкой корректности ввода.
- Вычисление дискриминанта и анализ его значения для определения числа действительных корней.
- Обработка случаев, когда действительных корней нет.

- Программа продемонстрировала корректную работу на тестовых данных, обеспечив точность вычислений и пользовательский ввод без ошибок.

Программа отвечает требованиям задания, обладает надежностью (обработка ошибок ввода) и удобным интерфейсом взаимодействия. Использование языка Rust позволило обеспечить высокую производительность и безопасность работы с данными.