



**Министерство науки и высшего образования Российской
Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана (национальный исследовательский
университет)» (МГТУ им. Н.Э. Баумана)**

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

**Отчёт по лабораторной работе №6
«Разработка на языке программирования Rust»**

Выполнил: Студент группы ИУ5-31Б – Толстолицкий Д.А.

Москва, 2022

Задание:

Разработать программу для решения биквадратного уравнения.

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

- Программа осуществляет ввод с клавиатуры коэффициентов A, B, C, вычисляет дискриминант и **ДЕЙСТВИТЕЛЬНЫЕ** корни уравнения (в зависимости от дискриминанта).
- 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.

Текст программы:

```
use std::io;

fn main() {
    println!("Hello, world!");
    for x in get_roots_vec(getkoeffs()) {
        if x > 0 as f64 {
            println!("{}", x.sqrt(), -x.sqrt())
        }
        else if x == 0 as f64 {
            println!("{}", x)
        }
    }
}

fn get_roots(a: i64, b: i64, c: i64) -> [f64; 2] {
    let mut result: [f64; 2] = [-999999999.9, -999999999.9];
    let d: f64 = (b * b - 4 * a * c) as f64;
    let mut root: f64 = 0.0;
    if d == 0.0 {
        root = -(b as f64) / (2.0 * (a as f64));
        result[0] = root;
    } else if d > 0.0 {
        let sq_d: f64 = d.sqrt();
        let root1 = -(b as f64) + sq_d / (2.0 * (a as f64));
        let root2 = -(b as f64) - sq_d / (2.0 * (a as f64));
        result[0] = root1;
        result[1] = root2;
    } else {}
    return result;
}

fn get_roots_vec(intvec: [i64; 3]) -> [f64; 2] {
    let a: i64 = intvec[0];
    let b: i64 = intvec[1];
    let c: i64 = intvec[2];
    let mut result: [f64; 2] = [-999999999.9, -999999999.9];
    let d: f64 = (b * b - 4 * a * c) as f64;
    let mut root: f64 = 0.0;
```

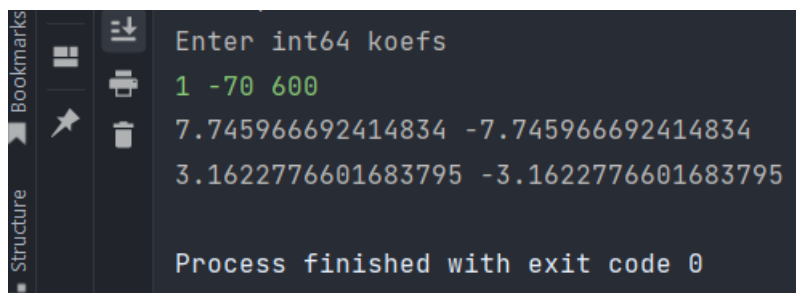
```

    if d == 0.0 {
        root = -(b as f64) / (2.0 * (a as f64));
        result[0] = root;
    } else if d > 0.0 {
        let sq_d: f64 = d.sqrt();
        let root1 = -(b as f64) + sq_d / (2.0 * (a as f64));
        let root2 = -(b as f64) - sq_d / (2.0 * (a as f64));
        result[0] = root1;
        result[1] = root2;
    } else {}
    return result;
}

// "1 2 3"
fn getkoeffs() -> [i64; 3] {
    loop {
        let mut input = String::new();
        println!("Enter int64 koeffs");
        io::stdin()
            .read_line(&mut input)
            .expect("Неверно введена строка");
        let vec = input.split(" ").collect::<Vec<&str>>();
        let mut intvec: [i64; 3] = [0, 0, 0];
        for i in 0..3 {
            let res_pars = (vec[i]).parse::<i64>();
            match res_pars {
                Ok(res_i64) => {intvec[i] = res_i64;}
                Err(_e) => {println!("ERROR")}
            }
        }
        return intvec;
    }
};
}

```

Результаты выполнения программы:



```

Enter int64 koeffs
1 -70 600
7.745966692414834 -7.745966692414834
3.1622776601683795 -3.1622776601683795

Process finished with exit code 0

```