Московский государственный технический университет им. Н.Э. Баумана

Факультет Радиотехнический
Кафедра “Системы обработки информации и управления”

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №1

Выполнил:
студент группы РТ5-31Б:
Длютров Тимофей
Олегович
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Юрий
Евгеньевич
Подпись и дата:

Москва, 2025 г.

Постановка задачи

Разработать консольное приложение на языке rust для решения биквадратного уравнения вида $Ax^4 + Bx^2 + C = 0$, которое должно принимать коэффициенты A, B, C либо через параметры командной строки, либо интерактивно с клавиатуры (если параметры не предоставлены), при этом обеспечивать проверку корректности ввода: при некорректных значениях (не преобразуемых в float) запрашивать повторный ввод до получения валидных данных. Программа вычисляет дискриминант и определяет все действительные корни уравнения (если они существуют), учитывая особенности биквадратной подстановки.

Текст программы

```rust
use std::io;
use std::io::Write;
use std::process;
use std::env;

fn error_exit(comment: &str) {
    println!("{}", comment);
    process::exit(-1);
}

fn read_line(comment: &str, is_on_one_line: bool) -> String {
    if is_on_one_line {
        print!("{}", comment);
        io::stdout().flush();
    }
    else {
        println!("{}", comment);
    }
    let mut string: String = String::new();
    io::stdin().read_line(&mut string)
        .ok()
        .expect("Error read line!");
    return string;
}

fn main() {
    let args: Vec<_> = env::args().collect();
    let mut need_hand_input = false;
    let mut num_inputs: [f64; 3] = [0.0, 0.0, 0.0];

    if args.len() > 3 {
        for i in 1..4 {
            let convertation_result = args[i].trim().parse::<f64>();
            match convertation_result {
                Ok(n) => num_inputs[i - 1] = n,
                Err(e) => { println!("\x1b[91mОшибка в консольных аргументах ({}-й, не конвертируется в число). Переход к ручному вводу.\x1b[0m", i); need_hand_input = true; break; }
            }
            if (i == 1 && num_inputs[0] == 0.0) {
                println!("\x1b[91mОшибка в консольных аргументах ({}-й, не может быть нулём). Переход к ручному вводу.\x1b[0m", i);
                need_hand_input = true;
                break;
            }
        }
    }
    else {
        println!("\x1b[91mОшибка в консольных аргументах (меньше, чем 3). Переход к ручному вводу.\x1b[0m");
        need_hand_input = true;
    }

    if need_hand_input {
        let premessages: [&str; 3] = ["Введите коэффициент A: ", "Введите коэффициент B: ", "Введите коэффициент C: "];
        let mut input_finished: bool = false;
        for i in 0..3 {
            let mut input_finished: bool = false;
            while !input_finished {
                let current_input = read_line(premessages[i], true);
                let convertation_result = current_input.trim().parse::<f64>();
                match convertation_result {
                    Ok(n) => num_inputs[i] = n,
                    Err(e) => { println!("\x1b[91mВедённое значение не является числом!\x1b[0m"); continue; }
                }
                if (i == 0 && num_inputs[0] == 0.0) {
                    println!("\x1b[91mПервый коэффициент не может быть нулём!\x1b[0m");
                    continue;
                }
                input_finished = true;
            }
        }
    }

    let d = num_inputs[1] * num_inputs[1] - 4.0 * num_inputs[0] * num_inputs[2];
    if d < 0.0 {
        println!("\x1b[91mДискриминант меньше нуля! Корней нет.\x1b[0m");
    }
    else if d == 0.0 {
        let c = -1.0 * num_inputs[1] / 2.0 / num_inputs[0];
        if c < 0.0 {
            println!("\x1b[91mДискриминант равен нулю. Корней нет.\x1b[0m");
        }
        else if c == 0.0 {
            println!("\x1b[92mДискриминант равен нулю. Единственный корень: {}.\x1b[0m", c);
        }
        else {
            println!("\x1b[92mДискриминант равен нулю. Корни: {}, {}.\x1b[0m", f64::sqrt(c), -1.0 * f64::sqrt(c));
        }
    }
    else {
        let c1 = (-1.0 * num_inputs[1] + f64::sqrt(d)) / 2.0 / num_inputs[0];
        let c2 = (-1.0 * num_inputs[1] - f64::sqrt(d)) / 2.0 / num_inputs[0];
        if c1 < 0.0 && c2 < 0.0 { println!("\x1b[91mДискриминант больше нуля. Корней нет.\x1b[0m"); process::exit(0); }
        print!("\x1b[92mДискриминант больше нуля. Корни: ");
        if c1 == 0.0 { print!("{} и ", c1); }
        if c1 > 0.0 { print!("{}, {} и ", f64::sqrt(c1), -1.0 * f64::sqrt(c1)); }
        if c2 == 0.0 { print!("{}", c2); }
        if c2 > 0.0 { print!("{}, {}", f64::sqrt(c2), -1.0 * f64::sqrt(c2)); }
    }
    println!("\x1b[0m");
}

```

Анализ результатов

Тест 1 (аргументы не вводятся):

```
Ошибка в консольных аргументах (меньше, чем 3). Переход к ручному вводу.
Введите коэффициент A: 0
Первый коэффициент не может быть нулём!
Введите коэффициент A: 1
Введите коэффициент B: sd
Ведённое значение не является числом!
Введите коэффициент B: -3
Введите коэффициент C: 2
Дискриминант больше нуля. Корни: 1.4142135623730951, -1.4142135623730951 и 1, -1
```

Тест 2 (Вводятся аргументы “a”, “2”, “3”):

```
Ошибка в консольных аргументах (1-й, не конвертируется в число). Переход к ручному вводу.
Введите коэффициент A: 1
Введите коэффициент B: 3
Введите коэффициент C: 2
Дискриминант больше нуля. Корней нет.
```

Тест 3 (Вводятся аргументы “2”, “2”):

```
Ошибка в консольных аргументах (меньше, чем 3). Переход к ручному вводу.
Введите коэффициент A: 1
Введите коэффициент B: 4
Введите коэффициент C: 4
Дискриминант равен нулю. Корней нет.
```

Тест 4 (Вводятся аргументы “2”, “a”, “3”):

```
Ошибка в консольных аргументах (2-й, не конвертируется в число). Переход к ручному вводу.
Введите коэффициент A: 1
Введите коэффициент B: -4
Введите коэффициент C: 4
Дискриминант равен нулю. Корни: 1.4142135623730951, -1.4142135623730951.
```

Тест 5 (Вводятся аргументы “1”, “0”, “0”):

```
Дискриминант равен нулю. Единственный корень: 0.
```

Тест 6 (Вводятся аргументы “1”, “-2”, “0”):

```
Дискриминант больше нуля. Корни: 1.4142135623730951, -1.4142135623730951 и 0
```

Тест 7 (Вводятся аргументы “1”, “2”, “3”):

```
Дискриминант меньше нуля! Корней нет.
```