Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»

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

Выполнил:
Студент группы ИУ5-32Б:
Арзамасцев Артем
Подпись и дата:

Проверил:
преподаватель каф. ИУ5
Гапанюк Ю.Е.
Подпись и дата:

Москва, 2022 г.

Описание задания

1. Реализуйте любое из заданий курса на языке программирования Rust.
2. Разработайте хотя бы один макрос.
3. Разработайте модульные тесты (не менее 3 тестов).

Текст программы

Для выполнения этой лабораторной работы, было выбрано задание первой лабораторной работы (решение биквадратного уравнения), только теперь оно решено на языке программирования Rust.

```
use std::io;
use std::env;
use std::mem;

#[derive(Debug, Copy, Clone, PartialEq)]
enum SquareRootResult {
    NoRoots,
    OneRoot(f64),
    TwoRoots { root1: f64, root2: f64 },
    ThreeRoots { root1: f64, root2: f64, root3:f64 },
    FourRoots { root1: f64, root2: f64, root3:f64, root4:f64 },
}

struct Equation {
    a: f64,
    b: f64,
    c: f64,
    diskr: f64,
    res: SquareRootResult,
}

impl Equation {
    fn calculate_roots(&mut self) {
        self.diskr = self.b.powi(2) - 4.0 * self.a * self.c;
        self.res = {
            if self.diskr < 0.0 {
                SquareRootResult::NoRoots
            } else if self.diskr == 0.0 {
                let rt = -self.b / (2.0 * self.a);
                if rt <= 0.0 {
                    SquareRootResult::NoRoots
                } else if rt == 0.0 {
                    SquareRootResult::OneRoot(rt)
                } else {
                    SquareRootResult::TwoRoots{root1: -rt.sqrt(), root2:
rt.sqrt()}
                }
```

```
            } else {
                let mut rt1 = (-self.b - self.diskr.sqrt()) / (2.0 * self.a);
                let mut rt2 = (-self.b + self.diskr.sqrt()) / (2.0 * self.a);
                if rt1 > rt2 {
                    mem::swap(&mut rt1, &mut rt2);
                }
                if rt1 * rt2 == 0.0 {
                    if rt2 > 0.0 {
                        SquareRootResult::ThreeRoots{root1: rt1, root2: -
rt2.sqrt(), root3: rt2.sqrt()}
                    } else {
                        SquareRootResult::OneRoot(rt2)
                    }
                }
                else if rt1 * rt2 < 0.0 {
                    SquareRootResult::TwoRoots{root1: -rt2.sqrt(), root2:
rt2.sqrt()}
                }
                else {
                    if rt1 > 0.0 {
                        SquareRootResult::FourRoots{root1: -rt1.sqrt(), root2:
rt1.sqrt(), root3: -rt2.sqrt(), root4: rt2.sqrt()}
                    } else {
                        SquareRootResult::NoRoots
                    }
                }
            }
        };
    }

    fn set_coef(&mut self) {
        let factors = read_factors();
        self.a = factors[0];
        self.b = factors[1];
        self.c = factors[2];
    }
}

fn read_factors() -> [f64; 3] {
    let mut factors:[f64; 3] = [0.0, 0.0, 0.0];
    let name_factors = ["a", "b", "c"];
    let mut curr_index = 0;
    let args: Vec<String> = env::args().collect();
    for i in args[2..].iter() {
        let row = String::from(i);
        match row.trim().parse() {
            Ok(val) => {
            factors[curr_index] = val;
            curr_index += 1;
            }
            Err(_) => {
```

```
                continue;
            }
        };
    }
    if curr_index > 2 {
        return factors;
    }
    while curr_index <= 2 {
        println!("введите коэффициент {}: ", name_factors[curr_index]);
        let mut input = String::new();
        io::stdin()
            .read_line(&mut input)
            .expect("Неверно введена строка");
        match input.trim().parse() {
            Ok(val) => {
                factors[curr_index] = val;
                curr_index += 1;
            }
            Err(_) => {
                continue;
            }
        }
    }
    return factors;
}

macro_rules! root_derivation {
    ($e:expr) => {{
        {
            use SquareRootResult::*;
            let eq: Equation = $e;
            let text_res = match eq.res {
                NoRoots => format!("Корней нет"),
                OneRoot(rt) => format!("Один корень => {}", rt),
                TwoRoots { root1, root2 } => format!("Два корня => {} и {}",
root1, root2),
                ThreeRoots { root1, root2, root3 } => format!("Три корня => {},
{} и {}", root1, root2, root3),
                FourRoots { root1, root2, root3, root4 } => format!("Четыре корня
=> {}, {}, {} и {}", root1, root2, root3, root4),
            };
            println!("{}", text_res);
        }
    }};
}

fn main() {
    let mut eq = Equation {
        a: 0.0,
        b: 0.0,
        c: 0.0,
```

```
        diskr: 0.0,
        res: SquareRootResult::NoRoots,
    };
    eq.set_coef();
    eq.calculate_roots();
    root_derivation!(eq);
}


#[test]
fn test_calc1() {
    use SquareRootResult::*;
    let mut eq = Equation {
        a: 1.24,
        b: 54.12,
        c: 5.0,
        diskr: 0.0,
        res: SquareRootResult::NoRoots,
    };
    eq.calculate_roots();
    assert_eq!(eq.res, NoRoots);
}

#[test]
fn test_calc2() {
    use SquareRootResult::*;
    let mut eq = Equation {
        a: 500.0,
        b: 5500.0,
        c: -23068.8,
        diskr: 0.0,
        res: SquareRootResult::NoRoots,
    };
    eq.calculate_roots();
    assert_eq!(eq.res, TwoRoots { root1: -1.8, root2: 1.8 });
}

#[test]
fn test_calc3() {
    use SquareRootResult::*;
    let mut eq = Equation {
        a: 2.0,
        b: -50.0,
        c: 0.0,
        diskr: 0.0,
        res: SquareRootResult::NoRoots,
    };
    eq.calculate_roots();
    assert_eq!(eq.res, ThreeRoots { root1: 0.0, root2: -5.0, root3: 5.0 });
}
```

Примеры выполнения программ

```
artem@LAPTOP-9N3DM2VF:/mnt/d/Course_BKIT_3sem/lab6$ cargo run main.rs 3 dsgg 6 sd dggg
    Finished dev [unoptimized + debuginfo] target(s) in 0.05s
     Running `target/debug/lab6 main.rs 3 dsgg 6 sd dggg`
введите коэффициент c:
5
Корней нет
artem@LAPTOP-9N3DM2VF:/mnt/d/Course_BKIT_3sem/lab6$
```

```
artem@LAPTOP-9N3DM2VF:/mnt/d/Course_BKIT_3sem/lab6$ cargo run main.rs
    Finished dev [unoptimized + debuginfo] target(s) in 0.08s
     Running `target/debug/lab6 main.rs`
введите коэффициент a:
1
введите коэффициент b:
-25
введите коэффициент c:
144
Четыре корня => -3, 3, -4 и 4
artem@LAPTOP-9N3DM2VF:/mnt/d/Course_BKIT_3sem/lab6$
```

```
artem@LAPTOP-9N3DM2VF:/mnt/d/Course_BKIT_3sem/lab6$ cargo test
   Compiling lab6 v0.1.0 (/mnt/d/Course_BKIT_3sem/lab6)
    Finished test [unoptimized + debuginfo] target(s) in 4.17s
     Running unittests src/main.rs (target/debug/deps/lab6-694038c5f0209411)

running 3 tests
test test_calc1 ... ok
test test_calc2 ... ok
test test_calc3 ... ok

test result: ok. 3 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

artem@LAPTOP-9N3DM2VF:/mnt/d/Course_BKIT_3sem/lab6$
```