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

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Лабораторная работа №6

по предмету

«Разработка комплексного приложения»

Выполнил:

студент группы № ИУ5-33Б

Комаров Дмитрий

Проверил:

Преподаватель кафедры ИУ-5

Гапанюк Юрий

2022 г.

Задание.

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

Код программы:

```rust
use std::io;

#[derive(Debug, Copy, Clone)]
///Тип решения квадратного уравнения
enum SquareRootResult {
    /// Unit-тип
    NoRoots,
    /// Один корень - кортежная структура
    OneRoot(f64),
    /// C-подобная структура 2 корня
    TwoRoots {root1: f64, root2: f64},
    ThreeRoots {root1: f64, root2: f64, root3: f64},
    /// 4 - корня
    FourRoots {root1: f64, root2: f64, root3: f64, root4: f64},
}

#[derive(Debug, Copy, Clone)]
/// Структура, соответствующая уравнению
struct Equation {
    /// Коэффициент A
    c_a: f64,
    /// Коэффициент B
    c_b: f64,
    /// Коэффициент C
    c_c: f64,
    /// Дискриминант
    diskr: f64,
    /// Корни
    res: SquareRootResult,
}

impl Equation {
    /// Функция вычисления корней
    fn calculate_roots(&mut self)->Vec<f64> {
        self.diskr = self.c_b.powi(2) - 4.0 * self.c_a * self.c_c;
        let mut values:Vec<f64>=Vec::new();
        if self.diskr == 0.0 {
            let rt = -self.c_b / (2.0 * self.c_a);
            if rt > 0.0 {
                values.push(rt.sqrt());
                values.push(-rt.sqrt());
            } else if rt == 0.0 {
                values.push(rt);
            }
        }
```

```
        else if self.diskr > 0.0 {
            let mut rt1 = (-self.c_b - self.diskr.sqrt()) / (2.0 * self.c_a);
            let mut rt3 = (-self.c_b + self.diskr.sqrt()) / (2.0 * self.c_a);
            if rt1 > 0.0 {
                rt1 = rt1.sqrt();
                values.push(rt1);
                values.push(-rt1);
            }else if rt1 == 0.0 {
                values.push(rt1);
            }
            if rt3 > 0.0 {
                rt3 = rt3.sqrt();
                values.push(rt3);
                values.push(-rt3);
            }else if rt3 == 0.0 {
                values.push(rt3)
            }
        }
        return values

    }


    /// Ввод одного коэффициента
    fn get_coef(message: &str) -> f64 {
        return loop {
            let mut input = String::new();
            println!("{}", message);
            io::stdin()
                .read_line(&mut input)
                .expect("Неверно введена строка");
            match input.trim().parse() {
                Ok(val) => {
                    break val;
                }
                Err(_) => {
                    continue;
                }
            }
        };
    }

    fn get_coefs(&mut self) -> () {
        self.c_a = Equation::get_coef("Введите коэффициент A: ");
        self.c_b = Equation::get_coef("Введите коэффициент B: ");
        self.c_c = Equation::get_coef("Введите коэффициент C: ");
    }
}

#[cfg(test)]
mod tests{
```

```rust
    use super::*;
    #[test]
    fn test1(){
        let mut eq = Equation {
            c_a: 1.0,
            c_b: -5.0,
            c_c: -36.0,
            diskr: 0.0,
            res: SquareRootResult::NoRoots,
        };
        eq.diskr = eq.c_b.powi(2) - 4.0 * eq.c_a * eq.c_c;
        let mut need:Vec<f64>=Vec::new();
        need.push(3.0);
        need.push(-3.0);
        assert_eq!(eq.calculate_roots(), need);
    }
    #[test]
    fn test2() {
        let mut eq = Equation {
            c_a: 1.0,
            c_b: -5.0,
            c_c: 4.0,
            diskr: 0.0,
            res: SquareRootResult::NoRoots,
        };
        eq.diskr = eq.c_b.powi(2) - 4.0 * eq.c_a * eq.c_c;
        let mut need:Vec<f64>=Vec::new();
        need.push(1.0);
        need.push(-1.0);
        need.push(2.0);
        need.push(-2.0);
        assert_eq!(eq.calculate_roots(), need);
    }
    #[test]
    fn test3(){
        let mut eq = Equation {
            c_a: -4.0,
            c_b: 16.0,
            c_c: 0.0,
            diskr: 0.0,
            res: SquareRootResult::NoRoots,
        };
        eq.diskr = eq.c_b.powi(2) - 4.0 * eq.c_a * eq.c_c;
        let mut need:Vec<f64>=Vec::new();
        need.push(2.0);
        need.push(-2.0);
        need.push(-0.0);
        assert_eq!(eq.calculate_roots(), need);
    }
}
```

```rust
fn main() {
    use SquareRootResult::*;
    let mut eq = Equation {
        c_a: 0.0,
        c_b: 0.0,
        c_c: 0.0,
        diskr: 0.0,
        res: SquareRootResult::NoRoots,
    };
    eq.get_coefs();
    let values_f:Vec<f64>=eq.calculate_roots();
    eq.res = {
        if values_f.len() == 2 {
            TwoRoots {
                root1: values_f[0],
                root2: values_f[1],
            }

        } else if values_f.len() == 1 {
            OneRoot(values_f[0])
        } else if values_f.len() == 3 {
            ThreeRoots {
                root1: values_f[0],
                root2: values_f[1],
                root3: values_f[2],
            }
        } else if values_f.len() == 4 {
            FourRoots {
                root1: values_f[0],
                root2: values_f[1],
                root3: values_f[2],
                root4: values_f[3],
            }
        } else {
            NoRoots
        }
    };
    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!("Два корня => {}, {} и {}",
root1, root2, root3),
        FourRoots { root1, root2, root3, root4 } => format!("Четыре корня => {},
{}, {}, {}", root1, root2, root3, root4),
    };
    println!("{}", text_res);
}
```

Результат выполнения программы:

```
userdead@DESKTOP-439N993:/mnt/c/Users/dimka/BKIT_Labs/lab6/rust_project/lab6$ cargo run
   Compiling lab6 v0.1.0 (/mnt/c/Users/dimka/BKIT_Labs/lab6/rust_project/lab6)
    Finished dev [unoptimized + debuginfo] target(s) in 1.01s
     Running `target/debug/lab6`
Введите коэффициент A:
2.9
Введите коэффициент B:
-3.8
Введите коэффициент C:
1
Четыре корня => 0.6038340211207747, -0.6038340211207747, 0.9724861451575147, -0.9724861451575147
```

```
userdead@DESKTOP-439N993:/mnt/c/Users/dimka/BKIT_Labs/lab6/rust_project/lab6$ cargo test
   Compiling lab6 v0.1.0 (/mnt/c/Users/dimka/BKIT_Labs/lab6/rust_project/lab6)
    Finished test [unoptimized + debuginfo] target(s) in 0.75s
     Running unittests src/main.rs (target/debug/deps/lab6-211ad0e32995750a)

running 3 tests
test tests::test1 ... ok
test tests::test2 ... ok
test tests::test3 ... ok

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