Московский государственный технический университет им. Н.Э. Баумана

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

Курс «Базовые компоненты интернет-технологий»

Отчет по домашней работе по курсу БКИТ

Выполнил:
студент группы ИУ5-34Б
Жданова Яна
Подпись и дата:

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

2022

Описание задания:

Альтернативный вариант домашнего задания №2. Разработка программы на языке программирования Rust.

Текст программы:

```
use std::io;

#[derive(Debug, Copy, Clone)]
pub enum RootsOfTheEquation {
    NoRoots,
    OneRoot (f64),
    TwoRoots {rt1: f64, rt2: f64},
    ThreeRoots {rt1: f64, rt2: f64, rt3: f64},
    FourRoots {rt1: f64, rt2: f64, rt3: f64, rt4: f64}
}

#[derive(Debug, Copy, Clone)]
pub struct Equation {
    a: f64,
    b: f64,
    c: f64,
    d: f64,
    result: RootsOfTheEquation
}
impl Equation {
    fn get_roots(&mut self) {
        self.d = self.b.powi(2) - 4.0 * self.a * self.c;
        self.result = {
            if (self.a == 0.0) && (self.b == 0.0) {
                RootsOfTheEquation::NoRoots
            } else if (self.a == 0.0 && self.c == 0.0) || ((self.c == 0.0) &&
(self.b == 0.0 || (self.b/self.a > 0.0))){
                RootsOfTheEquation::OneRoot(0.0)
            } else if self.a == 0.0{
                if self.c/self.b < 0.0{
                    RootsOfTheEquation::TwoRoots { rt1:
((self.c/self.b).abs().sqrt()), rt2: (-((self.c/self.b).abs().sqrt())) }
                } else {
                    RootsOfTheEquation::NoRoots
                }
            } else if (self.c == 0.0) && (self.b/self.a < 0.0){
                let sq_ba: f64 = (self.b / self.a).abs();
                RootsOfTheEquation::ThreeRoots { rt1: (0.0), rt2: (sq_ba.sqrt()),
rt3: (-sq_ba.sqrt()) }
            }else {
                if self.d == 0.0 {
                    let roots: f64 = (-self.b) / (2.0 * self.a);
                    RootsOfTheEquation::TwoRoots { rt1: (roots.sqrt()), rt2: (-
roots.sqrt()) }
                } else if self.d > 0.0{
```

```
                    let root1: f64 = (-self.b + self.d.sqrt()) / (2.0 * self.a);
                    let root2: f64 = (-self.b - self.d.sqrt()) / (2.0 * self.a);
                    if (root1 > 0.0) && (root2 < 0.0){
                        let roots1: f64 = root1.sqrt();
                        RootsOfTheEquation::TwoRoots { rt1: (roots1), rt2: (-
roots1) }
                    } else if (root2 > 0.0) && (root1 < 0.0){
                        let roots2: f64 = root2.sqrt();
                        RootsOfTheEquation::TwoRoots { rt1: (roots2), rt2: (-
roots2) }
                    } else if (root1 > 0.0) && (root2 > 0.0) {
                        let roots1: f64 = root1.sqrt();
                        let roots2: f64 = root2.sqrt();
                        RootsOfTheEquation::FourRoots { rt1: (roots1), rt2: (-
roots1), rt3: (roots2), rt4: (-roots2) }
                    } else {
                        RootsOfTheEquation::NoRoots
                    }
                } else {
                    RootsOfTheEquation::NoRoots
                }
            };
        }

        fn get_coef(message: &str) -> f64 {
            return loop {
                let mut input = String::new();
                println!("{}", message);
                io::stdin()
                    .read_line(&mut input)
                    .expect("Ошибка ввода аргумента. Введите корректное число.");
                match input.trim().parse() {
                    Ok(val) => {
                        break val;
                    }
                    Err(_) => {
                        continue;
                    }
                }
            };
        }

        fn get_coefs(&mut self) -> () {
            self.a = Equation::get_coef("Введите коэффициент A: ");
            self.b = Equation::get_coef("Введите коэффициент B: ");
            self.c = Equation::get_coef("Введите коэффициент C: ");
        }
}
```

```
macro_rules! panic_cs {
    ($a:expr, $b:expr) => {
        if $a == 0.0 && $b == 0.0{
            panic!{"Уравнение не было введено корректно!"}
        }
    };
}

macro_rules! beautiful_print {
    ($a:expr, $b:expr, $c:expr) => {
        if $a != 0.0 && $b!= 0.0 && $c!=0.0{
            if $b > 0.0 && $c > 0.0{
            println!("{}x^4 + {}x^2 + {} = 0", $a, $b, $c);
            } else if $b < 0.0 && $c > 0.0{
                println!("{}x^4 - {}x^2 + {} = 0", $a, $b.abs(), $c);
            } else if $b > 0.0 && $c < 0.0{
                println!("{}x^4 + {}x^2 - {} = 0", $a, $b, $c.abs());
            } else {
                println!("{}x^4 - {}x^2 - {} = 0", $a, $b.abs(), $c.abs());
            }
        }else if $a == 0.0 && $b != 0.0 && $c != 0.0{
            if $c > 0.0{
                println!("{}x^2 + {} = 0", $b, $c);
            } else {
                println!("{}x^2 - {} = 0", $b, $c.abs());
            }
        }else if $a != 0.0 && $b == 0.0 && $c != 0.0{
            if $c > 0.0{
                println!("{}x^4 + {} = 0", $a, $c);
            } else {
                println!("{}x^4 - {} = 0", $a, $c.abs());
            }
        }else {
            println!("Уравнение не введено корректно.");
        }
    };
}

fn main(){
    use RootsOfTheEquation::*;
    let mut our = Equation {
        a: 0.0,
        b: 0.0,
        c: 0.0,
        d: 0.0,
        result: RootsOfTheEquation::NoRoots,
    };
    our.get_coefs();
    panic_cs!(our.a, our.b);
    beautiful_print!(our.a, our.b, our.c);
    our.get_roots();
```

```
    let text_results = match our.result{
        NoRoots => format!("Нет корней"),
        OneRoot(rt) => format!("Один корень: {}", rt),
        TwoRoots{ rt1, rt2 } => format!("Два корня: {} и {}", rt1, rt2),
        ThreeRoots { rt1, rt2, rt3 } => format!("Три корня: {}, {} и {}", rt1,
rt2, rt3),
        FourRoots { rt1, rt2, rt3, rt4 } => format!("Четыре корня: {} и {}, {} и
{}", rt1, rt2, rt3, rt4)
    };
    println!("{}", text_results);
}
////////////////////////////////////////////////////////////////////////////////
////////////////////////////////
#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    #[should_panic]
    fn panic_test(){
        panic_cs!(0.0, 0.0);
    }

    #[test]
    fn test_result_one(){
        let mut test_eq= Equation {
            a: 6.0,
            b: 0.0,
            c: 0.0,
            d: 0.0,
            result: RootsOfTheEquation::NoRoots,
        };
        test_eq.get_roots();
        let flag: bool;
        let for_match: RootsOfTheEquation = test_eq.result;
        match for_match{
            RootsOfTheEquation::OneRoot(0.0) => flag = true,
            _ => flag = false,
        }
        assert!(flag)
    }

    #[test]
    fn test_result_two(){
        let mut test_eq= Equation {
            a: 0.0,
            b: -20.0,
            c: 5.0,
            d: 0.0,
            result: RootsOfTheEquation::NoRoots,
        };
```

```
        test_eq.get_roots();
        let flag: bool;
        let for_match: RootsOfTheEquation = test_eq.result;
        match for_match{
            RootsOfTheEquation::TwoRoots { rt1:0.5, rt2:-0.5} => flag = true,
            _ => flag = false,
        }
        assert!(flag)
    }

    #[test]
    fn test_result_three(){
        let mut test_eq= Equation {
            a: 1.0,
            b: -1.0,
            c: 0.0,
            d: 0.0,
            result: RootsOfTheEquation::NoRoots,
        };
        test_eq.get_roots();
        let flag: bool;
        let for_match: RootsOfTheEquation = test_eq.result;
        match for_match{
            RootsOfTheEquation::ThreeRoots { rt1:0.0, rt2:1.0, rt3:-1.0 } => flag
= true,
            _ => flag = false,
        }
        assert!(flag)
    }

    #[test]
    fn test_result_four(){
        let mut test_eq= Equation {
            a: 1.0,
            b: -5.0,
            c: 4.0,
            d: 0.0,
            result: RootsOfTheEquation::NoRoots,
        };
        test_eq.get_roots();
        let flag: bool;
        let for_match: RootsOfTheEquation = test_eq.result;
        match for_match{
            RootsOfTheEquation::FourRoots { rt1:2.0, rt2:-2.0, rt3:1.0, rt4:-1.0
} => flag = true,
            _ => flag = false,
        }
        assert!(flag)
    }

}
```

Результат выполнения программы:

```
PS C:\Users\user\Desktop\workinn on rust> cd lab_rust
PS C:\Users\user\Desktop\workinn on rust\lab_rust> cargo run
   Compiling lab_rust v0.1.0 (C:\Users\user\Desktop\workinn on rust\lab_rust)
    Finished dev [unoptimized + debuginfo] target(s) in 3.58s
     Running `target\debug\lab_rust.exe`
Введите коэффициент A:
1
Введите коэффициент B:
В случае ввода текста!
Введите коэффициент B:
Он будет зациклен до тех пор
Введите коэффициент B:
Пока не будет введено корректное число
Введите коэффициент B:
-5
Введите коэффициент C:
:))))
Введите коэффициент C:
4
1x^4 - 5x^2 + 4 = 0
Четыре корня: 2 и -2, 1 и -1
PS C:\Users\user\Desktop\workinn on rust\lab_rust> cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.01s
     Running `target\debug\lab_rust.exe`
Введите коэффициент A:
9
Введите коэффициент B:
17
Введите коэффициент C:
-2
9x^4 + 17x^2 - 2 = 0
Два корня: 0.3333333333333333 и -0.3333333333333333
PS C:\Users\user\Desktop\workinn on rust\lab_rust> cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.01s
     Running `target\debug\lab_rust.exe`
Введите коэффициент A:
10
Введите коэффициент B:
0
Введите коэффициент C:
3
10x^4 + 3 = 0
Нет корней
```

```
PS C:\Users\user\Desktop\workinn on rust\lab_rust> cargo run
   Compiling lab_rust v0.1.0 (C:\Users\user\Desktop\workinn on rust\lab_rust)
    Finished dev [unoptimized + debuginfo] target(s) in 1.25s
     Running `target\debug\lab_rust.exe`
Введите коэффициент A:
10
Введите коэффициент B:
-5
Введите коэффициент C:
0
10x^4 - 5x^2 = 0
Три корня: 0, 0.7071067811865476 и -0.7071067811865476
```

Результаты работы тестов:

```
running 5 tests
test tests::test_result_four ... ok
test tests::test_result_one ... ok
test tests::panic_test - should panic ... ok
test tests::test_result_three ... ok
test tests::test_result_two ... ok

test result: ok. 5 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s
```