Московский государственный технический университет имени Н.Э. Баумана

Факультет «Информатика и вычислительная техника»
Кафедра ИУ5 «Система обработки информации и управления»

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

Выполнил
студент ИУ5-35Б
Рябова С.А.

Проверил
Преподаватель каф. ИУ5
Гапанюк Ю.Е.

Москва 2024

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 программ

Main

```
use std::io;

macro_rules! get_coefficients {
    ($a:expr, $b:expr, $c:expr) => {
        println!("Enter coefficient A: ");
        let mut a_str = String::new();
        io::stdin().read_line(&mut a_str).unwrap();
        *$a = a_str.trim().parse().unwrap();

        println!("Enter coefficient B: ");
        let mut b_str = String::new();
        io::stdin().read_line(&mut b_str).unwrap();
        *$b = b_str.trim().parse().unwrap();

        println!("Enter coefficient C: ");
        let mut c_str = String::new();
        io::stdin().read_line(&mut c_str).unwrap();
        *$c = c_str.trim().parse().unwrap();
    };
}

fn get_roots(a: f64, b: f64, c: f64) -> Vec<f64> {
    let discriminant = b * b - 4.0 * a * c;
    let mut ys: Vec<f64> = Vec::new();
    if discriminant == 0.0 {
        ys.push(-b / (2.0 * a))
    } else if discriminant > 0.0 {
        ys.push((-b + discriminant.sqrt()) / (2.0 * a));
        ys.push((-b - discriminant.sqrt()) / (2.0 * a));
    }
    let mut roots: Vec<f64> = Vec::new();
    for i in ys {
```

```
        if i > 0.0 {
            roots.push(i.sqrt());
            roots.push(-i.sqrt());
        } else if i == 0.0 {
            roots.push(0.0);
        }
    }
    return roots;
}

#[cfg(test)]
mod tests {
    use super::*;

    #[test]
    fn test_one_root() {
        let a: f64 = 1.0;
        let b: f64 = 0.0;
        let c: f64 = 0.0;
        let mut v: Vec<f64> = vec![0.0];
        v.sort_by(|a, b| a.partial_cmp(b).unwrap());
        let mut res: Vec<f64> = get_roots(a, b, c);
        res.sort_by(|a, b| a.partial_cmp(b).unwrap());
        assert_eq!(res, v);
    }

    #[test]
    fn test_two_roots() {
        let a: f64 = 1.0;
        let b: f64 = -2.0;
        let c: f64 = 1.0;
        let mut v: Vec<f64> = vec![1.0, -1.0];
        v.sort_by(|a, b| a.partial_cmp(b).unwrap());
        let mut res: Vec<f64> = get_roots(a, b, c);
        res.sort_by(|a, b| a.partial_cmp(b).unwrap());
        assert_eq!(res, v);
    }
    #[test]
    fn test_four_roots() {
        let a: f64 = 1.0;
        let b: f64 = -5.0;
        let c: f64 = 4.0;
        let mut v: Vec<f64> = vec![1.0, -1.0, 2.0, -2.0];
        v.sort_by(|a, b| a.partial_cmp(b).unwrap());
        let mut res: Vec<f64> = get_roots(a, b, c);
```

```
        res.sort_by(|a, b| a.partial_cmp(b).unwrap());
        assert_eq!(res, v);
    }
}

fn main() {
    let mut a: f64 = 0.0;
    let mut b: f64 = 0.0;
    let mut c: f64 = 0.0;
    get_coefficients!(&mut a, &mut b, &mut c);
    let roots = get_roots(a, b, c);
    if roots.len() == 0 {
        println!("No real roots");
    } else if roots.len() == 1 {
        println!("One real root: {}", roots[0]);
    } else if roots.len() == 2 {
        println!("Two real roots: {} and {}", roots[0], roots[1]);
    } else if roots.len() == 4 {
        println!(
            "Four real roots: {}, {}, {}, {}",
            roots[0], roots[1], roots[2], roots[3]
        )
    }
}
```

Экранные формы выполнения программ

```
Enter coefficient A:
1
Enter coefficient B:
0
Enter coefficient C:
-1
Two real roots: 1 and -1

=== Code Execution Successful ===
```

```
Enter coefficient A:
1
Enter coefficient B:
1
Enter coefficient C:
1
No real roots

=== Code Execution Successful ===
```

```
Enter coefficient A:
1
Enter coefficient B:
-5
Enter coefficient C:
4
Four real roots: 2, -2, 1, -1

=== Code Execution Successful ===
```