**Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технический университет**

**имени Н.Э. Баумана (национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра «Системы обработки информации и управления»**

**Курс «Базовые компоненты интернет-технологий»**

Отчёт по лабораторной работе №6

## «[Разработка на языке программирования Rust](https://github.com/ugapanyuk/BKIT_2022/wiki/lab_rust)»

Выполнил:

студент группы ИУ5-33Б

Рыбин Владислав

Проверил:

к.т.н., доц., Ю. Е. Гапанюк

2022 г.

Биквадратное уравнение

Текст программы

use std::io;  
  
#[derive(Debug,Clone, Copy)]  
enum Roots {  
  
 *NoRoots*,  
 *OneRoot*(f32),  
 *TwoRoots*{ root1: f32, root2 : f32 }  
  
}  
  
#[derive(Debug,Clone, Copy)]  
enum Bi\_Roots{  
 *NoRoots*,  
 *TwoRoots* {  
 root1: f32,  
 root2: f32,  
 },  
 *FourRoots*{  
 root1: f32,  
 root2: f32,  
 root3: f32,  
 root4: f32,  
 }  
}  
#[derive(Debug,Clone, Copy)]  
struct Square\_Equation {  
 a : f32,  
 b: f32,  
 c: f32,  
 D: f32,  
 result: Roots  
  
}  
  
impl Square\_Equation {  
 fn calculate\_roots(&mut self)  
 {  
 self.D=self.b.powf(2.0)-4.0\*self.a\*self.c;  
 self.result = {  
 if self.D < 0.0 {  
 Roots::*NoRoots* } else if self.D == 0.0 {  
 let root: f32 = (-self.b) / (2.0 \* self.a);  
 Roots::*OneRoot*(root)  
 }  
 else {  
 let r1: f32 = (-self.b - self.D.powf(0.5)) / (2.0 \* self.a);  
 let r2: f32 = (-self.b + self.D.powf(0.5)) / (2.0 \* self.a);  
 Roots::*TwoRoots* { root1: r1, root2: r2 }  
  
 }  
 }  
}  
  
  
  
 fn get\_coefs(&mut self)  
 {  
 self.a=Square\_Equation::get\_coef("Введите коеффициент a:");  
 self.b=Square\_Equation::get\_coef("Введите коеффициент b:");  
 self.c=Square\_Equation::get\_coef("Введите коеффициент c:");  
 }  
 fn *get\_coef*(message: &str) -> f32  
 {  
  
  
 return loop {  
 let mut res= String::new();  
  
 println!("{}", message);  
 io::stdin()  
 .read\_line(&mut res)  
 .expect("Неверно введена строка");  
 match res.trim().parse() {  
 Ok(val) => {  
 break val  
 }  
 Err(\_)=> {  
 continue;  
 }  
 }  
  
 }  
 }  
}  
  
  
struct Bi\_Square\_Equation{  
 rts: Square\_Equation,  
 result: Bi\_Roots  
}  
  
impl Bi\_Square\_Equation {  
  
 fn get\_bi\_coefs(&mut self)  
 {  
  
 self.result = match self.rts.result{  
 Roots::*NoRoots* => Bi\_Roots::*NoRoots*,  
 Roots::*OneRoot*(rt) => Bi\_Roots::*TwoRoots* { root1: -rt.powf(0.5), root2: rt.powf(0.5) },  
 Roots::*TwoRoots* { root1, root2 } =>  
 Bi\_Roots::*FourRoots* { root1: -root1.powf(0.5), root2:root1.powf(0.5), root3:-root2.powf(0.5), root4: root2.powf(0.5)}  
 }  
  
  
 }  
}  
  
  
  
  
fn main() {  
  
  
 let mut eq = Square\_Equation {  
 a : 0.0,  
 b: 0.0,  
 c: 0.0,  
 D: 0.0,  
 result: Roots::*NoRoots* };  
  
 eq.get\_coefs();  
 eq.calculate\_roots();  
 let mut bi\_eq: Bi\_Square\_Equation = Bi\_Square\_Equation { rts: eq  
 , result: Bi\_Roots::*NoRoots* };  
  
  
 println!("a: {}, b: {}, c: {}",eq.a,eq.b,eq.c);  
 println!("Result: {:?}",eq.result);  
 let text\_res = match eq.result {  
 Roots::*NoRoots* => format!("Корней нет"),  
 Roots::*OneRoot*(rt) => format!("Один корень => {}", rt),  
 Roots::*TwoRoots* { root1, root2 } => format!("Два корня => {} и {}", root1, root2),  
 };  
 println!("{}\n\n", text\_res);  
  
  
 println!("Result: {:?}",bi\_eq.result);  
 let text\_res = match bi\_eq.result {  
 Bi\_Roots::*NoRoots* => format!("Корней нет"),  
 Bi\_Roots::*TwoRoots* { root1, root2 } => format!("Два корня => {} и {}", root1, root2),  
 Bi\_Roots::*FourRoots* { root1, root2 ,root3 ,root4 } => format!("4 корня => {} и {}, {} и {} ", root1, root2,root3,root4),  
 };  
 println!("{}", text\_res);  
  
  
}

Результаты работы программы



