Języki programowania i GUI

Lista 2 - 2025

1. Uzupełnij miejsca wykropkowane tak, aby poniższy kod działał prawidłowo.
```
use std::f64::consts::PI;
use Fig::*;

#[derive(Debug)]
enum Fig {Koło {r: f64 },Prost {a: f64, b: f64 },
       Kwadr {a: f64 },Romb {a: f64, alfa: f64 },}

fn pole(f: &Fig) -> f64 {...}
fn obwód(f: &Fig) -> f64 {...}
fn obrót90(f: &mut Fig) -> f64 {...}

fn main() {
    let figury = [Koło { r: 1.5 },Prost { a: 1.0, b: 2.0 },
                  Kwadr { a: 5.0 },Romb {a: 3.0,alfa: PI / 3.0,},
                 ];

    for f in &figury {
        println!("{f:?} ma pole={} obwód={}", pole(f), obwód(f))
        obrót90(&mut f);
        println!(" Po obroacie {f:?}")
    }
}
```
2. Dla struktury z poprzedniego zadania zaimplementuj trait `std::Display`.

3. Dla struktury `Fig` napisz funkcję `fn save(filename:&str,v:Vec<Fig>)`pozwalającą na zapis wektora Figur do pliku, oraz funkcję `fn load(filename:&str)-> Vec<Fig>)`, pozwalającą na odtworzenie zawartości wektora figur z pliku. Po wykonaniu kodu:
`save("Figury.txt",figury); let figury1=load("Figury.txt");`
wektory `figury` oraz `figury1` powinny być identyczne.

4. Wzbogać swoje rozwiązanie o obsługę błędów za pomocą `Result`.

5. Dla klasy `struct Frac(i32,i32)` zaimplementuj traity `Add`, `Sub`, `Mul`, `Div`, oraz dodaj `#[derive(Debug)]` oraz metodę `fn uprosc(&mut self)->Frac`, aby działało:
```
fn main(){
 let a=Frac(2,3); let b=Frac(2,4); let c=Frac(2,3); let d=(a+b-c)*b/c;
 println!("Wynik: {:?}",d.uprosc());
}
```
6. Zdefiniuj strukturę `Poly` w następujący sposób:

```
#[derive(Clone)]
struct Poly {
 a:Vec<f32> // wektor współczynników wielomianu
}
```

   Następnie zdefiniuj metodę `eval` pozwalającą wyliczyć wartość wielomianu w punkcie `x` oraz zaimplementuj traity `std::ops::Add`, `std::ops::Mul` oraz `std::ops::Sub`, tak aby możliwe było dodawanie, odejmowanie i mnożenie wielomianów przez wielomian oraz przez liczby z lewej i prawej strony.

7. Zaimplementuj dla wielomianów trait `std::fmt::Display`.