Системне Програмування

З використанням мови програмування **Rust.**
Fundamentals. Compound types. Coproducts

Складні типи

Інколи, (дуже часто) не можливо виразити логіку програми за допомогою стандартних типів, таких як:
i8, i16, i32, i64, i128, u8, u16, u32, u64, u128, f32, f64
unit
[]
String, &str,
Tuple,
Named Tuple,
Struct

Приклади такого коду...

```rust
fn min(xs: &[i32]) -> i32
```

Приклади такого коду...

```
fn index_of(x: i32, xs: &[i32]) -> usize
```

Приклади такого коду...

```
fn read_file(name: String) -> String
```

Приклади такого коду...

```rust
fn whatever(color: &str) { todo!() }

fn test() {
    whatever( color: "red");
    whatever( color: "Red");
    whatever( color: "RED");
    whatever( color: "yellow");
    whatever( color: "YELLOW");
    whatever( color: "Green");
    whatever( color: "Greeen");
}
```

скільки існує варіантів помилитися і як буде виглядати наш код...

Або...

$$ax^2 + bx + c = 0$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
fn solve_quadratic1(a: f32, b: f32, c: f32) -> ???
```

скільки існує варіантів? Якій тип має повернути функція?

Enum (Enumeration)

В Rust, enum — це тип, який дозволяє визначити набір варіантів (значень), кожен з яких може мати різну структуру та типи даних. Він широко використовується для опису даних, що можуть бути різних типів або станів, і дозволяє спростити роботу з ними за допомогою паттерн-матчингу.

Приклади використання Enum

```rust
enum Color {
    Red,
    Yellow,
    Green,
}

fn whatever1(color: Color) {}
```

```rust
fn test() {
    whatever1(Color::Green);
    whatever1(Color::Blue);
}
```

Приклади використання Enum

```
enum Switch {
    On,
    Off,
}

enum Option1 {
    Enabled,
    Disabled,
}
```

Приклади використання Enum

```
enum Message {
    Quit,
    Move { x: i32, y: i32 },
    Write(String),
    ChangeColor(i32, i32, i32),
}
```

Приклади такого коду...

```rust
fn min(xs: &i32) -> i32
```

```rust
enum Option<A> {
    None,
    Some(A),
}

fn min(xs: &i32) -> Option<i32>
```

Приклади такого коду...

```rust
fn index_of(x: i32, xs: &[i32]) -> usize
```

```rust
enum FoundResult<A> {
    NotFound,
    Found(A),
}

fn index_of(x: i32, xs: &[i32]) -> FoundResult<usize>
```

Приклади такого коду...

```
fn read_file(name: String) -> String
```

```
enum IOResult<E, A> {
    Error(E),
    Result(A)
}

fn read_file(name: String) -> IOResult<String, String>
```

Приклади такого коду...

```rust
fn whatever(color: &str) { todo!() }

fn test() {
    whatever( color: "red");
    whatever( color: "Red");
    whatever( color: "RED");
    whatever( color: "yellow");
    whatever( color: "YELLOW");
    whatever( color: "Green");
    whatever( color: "Greeen");
}
```

```rust
enum Color {
    Red,
    Yellow,
    Green,
}

fn whatever(color: Color)

fn test() {
    whatever(Red);
    whatever(Green);
    whatever(Blue);
}
```

Приклади використання Enum

```
enum AuthenticationResult {
    Token(String),
    InvalidUsername,
    InvalidPassword,
    PasswordExpired,
    UserIsLocked,
}

fn auth(username: String, password: String) -> AuthenticationResult
```

Квадратне рівняння

$$ax^2 + bx + c = 0$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
fn solve_quadratic1(a: f32, b: f32, c: f32) -> ???
```

- $b^2 - 4ac < 0$ - No roots
- $b^2 - 4ac = 0$ - One Root
- $b^2 - 4ac > 0$ - Two Roots

Квадратне рівняння

- $b^2 - 4ac < 0$ - No roots
- $b^2 - 4ac = 0$ - One Root
- $b^2 - 4ac > 0$ - Two Roots

```
enum Solution {
    NoRoots,
    OneRoot(f32),
    TwoRoots(f32, f32),
}
```

Квадратне рівняння

```rust
enum Solution {
    NoRoots,
    OneRoot(f32),
    TwoRoots(f32, f32),
}
```

```rust
fn solve_quadratic(a: f32, b: f32, c: f32) -> Solution {
    use Solution::*;

    let d :f32 = b.powi( n: 2) - 4.0 * a * c;

    if d < 0. {
        NoRoots
    } else if d == 0. {
        OneRoot(-b / (2. * a))
    } else {
        let dq :f32 = f32::sqrt(d);
        let a2 :f32 = a * 2.;
        TwoRoots((-b - dq) / a2, (-b + dq) / a2)
    }
}
```

**Код з лекцій,
презентації Keynote,
PDF-файли
знаходяться на GitHub:**

https://github.com/djnzx/rust-course