Системне Програмування

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

Олексій Рихальський

26.09.2024

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

Приклади такого коду 1

```rust
fn print_user(name: String, age: i32) {
    todo!()
}
```

```rust
fn test0a() {
    let name : &str = "Jim";
    let age : i32 = 33;

    print_user(name.to_string(), age);
}
```

Приклади такого коду 1

```rust
fn deliver_pizza(name: String, size: i32) {
    todo!()
}
```

```rust
fn test0b() {
    let pizza : &str = "Margarita";
    let size : i32 = 60;

    deliver_pizza(pizza.to_string(), size);
}
```

Приклади такого коду 1

```
let name : &str = "Jim";
let age : i32 = 33;

let pizza : &str = "Margarita";
let size : i32 = 60;

print_user(pizza.to_string(), age);
deliver_pizza(name.to_string(), size);

print_user(name.to_string(), size);
deliver_pizza(pizza.to_string(), age);
```

Один з варіантів вирішення

```
let person : (String, i32) = ("Jim".to_string(), 33);
let pizza : (String, i32) = ("Margarita".to_string(), 60);

print_user2(person);
deliver_pizza2(pizza);
```

```
fn print_user2(person: (String, i32)) { todo!() }

fn deliver_pizza2(pizza: (String, i32)) { todo!() }
```

Один з варіантів вирішення

```
let person : (String, i32) = ("Jim".to_string(), 33);
let pizza : (String, i32) = ("Margarita".to_string(), 60);

print_user2(person);
deliver_pizza2(pizza);
```

```
print_user2(pizza);
deliver_pizza2(person);
```

```
fn print_user2(person: (String, i32)) { todo!() }

fn deliver_pizza2(pizza: (String, i32)) { todo!() }
```

Tuple. Створення

```
let x : (i32, f64, String, bool) =
    (1, 3.14, "Alex".to_string(), true);
```

Може включати довільну кількість елементів довільного типу

Tuple. Доступ до елементів

```
let x : (i32, f64, String, bool) =
    (1, 3.14, "Alex".to_string(), true);

let a : i32 = x.0;
let b : f64 = x.1;
let c : String = x.2;
let d : bool = x.3;
```

Tuple. Деструктуризація

Деструктуризація - це одночасний доступ до всіх елементів та розбирання на составляючі.

```
let x : (i32, f64, String, bool) =
    (1, 3.14, "Alex".to_string(), true);

let (a : i32, b : f64, c : String, d : bool) = x;
```

Часто використовується коли Tuple прийшов як параметр

Tuple. Проблеми

```
let person : (String, i32) =
    ("Quattroformagio".to_string(), 33);

let pizza : (String, i32) =
    ("Jim".to_string(), 60);
```

Named Tuple.

```
struct User(String, i32);
struct Pizza(String, i32);

let person: User = User("Jim".to_string(), 33);
let pizza: Pizza = Pizza("Margarita".to_string(), 60);
```

Кількість елементів довільна
Тип елементів довільний

Named Tuple. Доступ до елементів

```
struct User(String, i32);
struct Pizza(String, i32);

let person: User = User("Jim".to_string(), 33);
let pizza: Pizza = Pizza("Margarita".to_string(), 60);

let x :String = person.0;
let y :i32 = person.1;
```

... не дуже зручно

Named Tuple. Деструктуризація

```
let person: User = User("Jim".to_string(), 33);

let User(name: String, age: i32) = person;
```

Named Tuple. Використання

```
let person: User = User("Jim".to_string(), 33);
let pizza: Pizza = Pizza("Margarita".to_string(), 60);

print_user(person);
deliver_pizza(pizza);

// will not compile
// print_user(pizza);
// deliver_pizza(person);
```

Big Profit ! ! !

Named Tuple. Проблеми

```
struct Point2d(i32, i32);

let p1 = Point2d(2, 3);
let p2 = Point2d(3, 2);

let Point2d(x :i32, y :i32) = p1;
let Point2d(y :i32, x :i32) = p2;
```

Легко помилитися, якщо дані однакові, однакового типу, майже однакової семантики

Struct. Декларація

```
struct Person {
    name: String,
    age: i32,
}
```

Кількість елементів довільна
Тип елементів довільний

Struct. Створення

```
let person: Person = Person {
    name: "Jim".to_string(),
    age: 33,
};
```

Struct. Доступ до елементів

```
let person: Person = Person {
    name: "Jim".to_string(),
    age: 33,
};
```

```
let name :String = person.name;
let age :i32 = person.age;
```

Struct. Деструктуризація

```
let person: Person = Person {
    name: "Jim".to_string(),
    age: 33,
};
```

```
let Person { name : String , age : i32 } = person;
```

```
let Person { surname, age : i32 } = person;
```

Struct. Помилитися складніше

```
struct Point2d {
    x: i32,
    y: i32,
}

let p1 = Point2d { x: 2, y: 3 };
let p2 = Point2d { x: 3, y: 2 };
```

Wrap-Up. Tuple

```
let x : (i32, f64, String, bool) =
    (1, 3.14, "Alex".to_string(), true);
let (a : i32, b : f64, c : String, d : bool) = x

let a : i32 = x.0;
let b : f64 = x.1;
let c : String = x.2;
let d : bool = x.3;
```

Wrap-Up. Named Tuple

```
struct Person(String, i32);

let person: Person = Person("Jim".to_string(), 33);

let Person(name: String, age: i32) = person.clone();

let n: String = person.0;
let a: i32 = person.1;
```

Wrap-Up. Struct

```rust
struct Pizza {
    name: String,
    size: i32,
}

let pizza: Pizza = Pizza {
    name: "Margarita".to_string(),
    size: 60,
};

let Pizza { name: String, size: i32 } = pizza

let n: String = pizza.name;
let s: i32 = pizza.size;
```

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

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