Санкт-Петербургский государственный университет

Кафедра системного программирования

Типы в Rust и прочее

Дмитриевцев Алексей Сергеевич

Санкт-Петербург
2024

Как в Rust дела с типами?

Rust:

- Статически типизированный
- Скалярные типы
- Составные типы
- Структуры

Скалярные типы

- Целые числа (u8, i32, ***usize***, etc.)
- Числа с плавающей запятой (f32, f64

Прописаны в IEE-754)

- Логические (bool)
- Символьный (char - символ в кодировке

UTF-32, 32 бита)

usize/isize

- Архитектура Фон-Неймана ???
- Архитектурозависимый

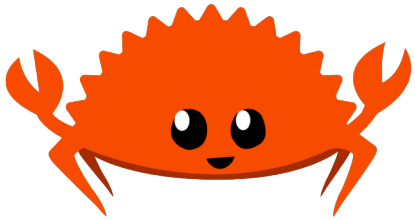
Для 32 битных систем usize = u32,
для 64 битных usize = u64.

Аналоги - size_t в C/C++, uint в Go.

Составные типы - Кортеж

Кортеж - универсальный способ объединения нескольких значений с различными типами в один составной тип. *Вспомним о кортежах, когда речь зайдет о структурах.*

```
fn main() {
let tup: (i32, f64, u8) = (500, 6.4, 1);
}
```



Аннотации к типам

```
let guess = "52".parse().expect("Not a number!");
```

```
let guess = "52".parse().expect("Not a number!");
    ^^^^^      ----- type must be known at this point
```

```
let guess: i32 = "52".parse().expect("Not a number!");
```

```
let guess = "52".parse::<i32>().expect("Not a number!");
```

Конструкция ::<some_type> называется turbofish и является проявлением параметрического полиморфизма, но об этом на следующих лекциях

Переполнение

- Дополнительный код
- А разве такое вообще возможно в Rust без unsafe?

При компиляции с флагом --release Rust не делает проверку на целочисленное переполнение

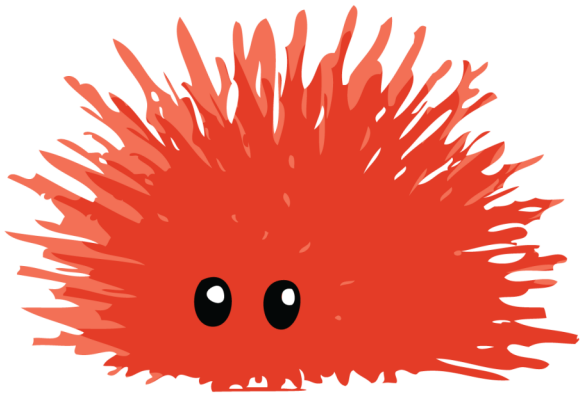
u8: 256 → 0, 257 → 1 и так далее

Приведение типов

- Явное (+)
- Неявное (-)

```
a = b as i32
```

- i8 → i32?
- u8 → i32?
- i32 → u32?
- u32 → i32?
- i128 → u8?
- char → u8?
- &str → i32?



Интересные особенности

- Синонимы типов

```
type amount = i32;

let x: i32 = 5;
let y: amount = 5;
```

Обычно используется в таких случаях:

```
type Thunk = Box<dyn Fn() + Send + 'static>;
```

Интересные особенности

- Специальные типы Unit Never

```
fn foo() -> ! {
    // тело функции
} // experimental

fn foo() -> () {
    // тело функции
}

fn foo() {
    // тело функции
}
```

Коллекции - vec

- Вектор, он же динамический массив, он же growable array, он же vec для Rust
- Capacity и reallocation
- Увеличивает размер при реаллокации в 2 раза, в других ЯП это не так. Ее нельзя переопределить.

Структуры

```
struct User {
    active: bool,
    username: String,
    email: String,
    sign_in_count: u64,
}


struct AlwaysEqual; //равносильно ()
```

Перечисления (enum)

```
enum Colour {
    red,
    black,
}

let four = Colour::red;
```

Перечисления (enum)

```
enum IpAddrKind {
    V4,
    V6,
}
```

```
struct IpAddr {
    kind: IpAddrKind,
    address: String,
}
```

```
let home = IpAddr {
    kind: IpAddrKind::V4,
    address: String::from("127.0.0.1"),
};
```

Перечисления (enum)

```
enum IpAddr {
    V4(String),
    V6(String),
}

let home = IpAddr::V4(String::from("127.0.0.1"));;
```

Коллекции - HashMap

- Использует хэш (удивительно)
- Хранит объекты в виде пары K - V (Ключ - Значение)
- Аналог словарей в питоне, но:

```
let mut scores = HashMap::new();
scores.insert(1, 10);
scores.insert(2, 50);
scores.insert("key_2", 50);
```

Что дальше?

В stdlib Rust'a есть много всего интересного:

- Box<T>
- Option<T> (перекочевало из ML-подобных языков)
- Result<T, E>
- Etc.
- Но об этом на других занятиях/курсах

Структуры данных

Какие структуры данных вы знаете?

- Стек
- Дерево
- Очередь
- Список
- Множество

Все они есть в collections::name, ***HO***

Ваши отзывы