Rust: управление сложностью кода

Пакеты

Пакетный менеджер Cargo, который позволяет создавать, тестировать и совместно использовать упаковки.

Упаковки

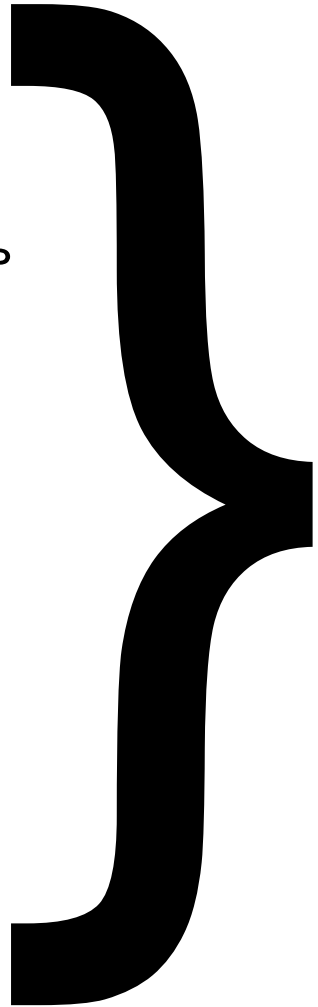
Дерево модулей, которое производит библиотеку или исполняемый файл.

Модули и use

Позволяют управлять организацией, областью видимости и приватностью путей.

Пути

Способ именования элемента, такого как структура, функция или модуль.



Модульная система

Пакеты (Packages):

Пакет в Rust - это каталог, который содержит файл Cargo.toml, описывающий метаданные вашего проекта, и один или более модулей. Пакет может представлять собой приложение или библиотеку.

Модули (Modules):

Модули - это способ организации кода внутри пакета. Они позволяют разбивать код на логически связанные блоки. Модуль может содержать функции, структуры, типы и другие элементы кода.

Модули могут быть вложенными, что позволяет создавать иерархию организации кода.

rust

```rust
mod my_module {
    // Содержимое модуля
}
```

Упаковки (Crates):

Упаковка в Rust - наименьшая единица компиляции в языке. Это фактически библиотека или исполняемый файл, который можно компилировать и связывать независимо.

Крейты можно рассматривать как отдельные модули кода, которые можно использовать и переиспользовать внутри или вне пакетов.

Каждый крейт обычно содержится в собственном каталоге и имеет собственный файл lib.rs или main.rs, который служит точкой входа.

Ключевое слово use:

Ключевое слово use используется для импорта элементов (функций, структур, типов и др.) из модулей или упаковок в текущую область видимости. Это упрощает доступ к элементам и позволяет избежать полной квалификации пути.

rust

```rust
use my_module::my_function;
```

Пути (Paths):

Путь в Rust - это способ указать местоположение элемента в иерархии модулей и упаковок. Пути могут быть абсолютными или относительными. Абсолютный путь начинается с имени упаковки корня, а относительный начинается с текущего контекста.

Пример абсолютного пути:

rust

```rust
use crate::my_module::my_function;
```

Пример относительного пути:

rust

```rust
use super::my_function;
```

Пути (Paths):

- Абсолютный путь начинается с корня упаковки, используя имя упаковки либо литерал crate.
- Относительный путь начинается с текущего модуля и использует self, super или идентификатор в текущем модуле.

```
mod front_of_house {
    mod hosting {
        fn add_to_waitlist() {}
    }
}

pub fn eat_at_restaurant() {
    // Абсолютный путь
    crate::front_of_house::hosting::add_to_waitlist();

    // Относительный путь
    front_of_house::hosting::add_to_waitlist();
}
```

Пути (Paths):

- Мы можем ввести путь в область видимости один раз, а затем вызывать элементы в этом пути, как если бы они были локальными элементами, с помощью ключевого слова use:

```
mod front_of_house {
    pub mod hosting {
        pub fn add_to_waitlist() {}
    }
}

use crate::front_of_house::hosting;

pub fn eat_at_restaurant() {
    hosting::add_to_waitlist();
    hosting::add_to_waitlist();
    hosting::add_to_waitlist();
}
```

Пути (Paths):

- Добавление use и пути в область видимости аналогично созданию символической ссылки в файловой системе.
- Благодаря добавлению use crate::front_of_house::hosting в корень упаковки модуль hosting сейчас является допустимым именем в этой области видимости, как будто модуль hosting был определен в корне упаковки.
- Указание относительного пути с помощью use немного отличается. Вместо того, чтобы начинать с имени в текущей области видимости, мы должны начинать путь, переданный инструкции use, с ключевого слова self.

```
mod front_of_house {
    pub mod hosting {
        pub fn add_to_waitlist() {}
    }
}

use self::front_of_house::hosting;

pub fn eat_at_restaurant() {
    hosting::add_to_waitlist();
    hosting::add_to_waitlist();
    hosting::add_to_waitlist();
}
```

Пространства имен

- В Rust, пространства имен реализованы с помощью ключевого слова use и модулей.
- В Rust, элементы по умолчанию являются приватными и не доступны извне модуля.
- Чтобы сделать элементы доступными за пределами модуля, используется ключевое слово pub.