ORGANIZER: GFI VBI | POWERED BY: WEB3HACKFEST | DEV PARTNER: CODE mely VU NGUYEN {C<>[)≡R}

RUST DEVELOPER BOOTCAMP

19:30 - 21:00 | 03/07/2023

Discord, Zoom Online

Xử lý lỗi và macro trong Rust

1. Xử lý lỗi

Mục đích của Xử lý lỗi

+ Tìm kiếm, debug, xử lý lỗi trong suốt quá trình thực hiện chương trình code
+ Thông báo lỗi cho người sử dụng khi người sử dụng ứng dụng thực hiện sai hành động

1. Xử lý lỗi

Có 2 loại lỗi :

+ Lỗi không thể phục hồi (Non-recoverable errors)
+ Lỗi có thể phục hồi (Recoverable errors)

1.1 Lỗi không thể phục hồi

panic: Dừng lại chương trình ngay tại thời điểm thực thi câu lệnh panic
assert: Kiểm tra biểu thức true hoặc false, nếu true tiếp tục thực hiện câu lệnh, nếu false dừng lại chương trình

```
fn main() {
    panic!("error!");
    println!("Never reached");
}
```

1.1 Lỗi không thể phục hồi

Ưu điểm:

+ Kiểm soát lỗi
+ Bảo mật

Nhược điểm:

+ Dừng chương trình
+ Không thể khôi phục lỗi

1.2 Lỗi có thể phục hồi

Lỗi để xử lý sẽ được trả về bằng Result hoặc Option

```
pub enum Result<T, E> {
    Ok(T),
    Err(E),
}

pub enum Option<T> {
    None,
    Some(T),
}
```

1.2 Trả về Error kiểu String

```
fn foo(number: i64) -> Result<(), String> {
  if number == 42 {
    return Ok(());
  }
  return Err("Wrong answer".to_string());
}
```

1.2 Trả về Error kiểu str

```
fn foo(number: i64) -> Result<(), &’static str > {
  if number == 42 {
    return Ok(());
  }
  return Err("Wrong answer");
}
```

1.2 Trả về Error kiểu enum

```
pub enum Error {
  WrongAnswer,
  More,
  Less,
}
```

```
fn foo(number: i64) -> Result<(), Error> {
  if number == 42 {
    return Ok(());
  }
else if number <= 41 {
      return Err(Error::More);
}
else if number >=43 {
      return Err(Error::Less);
}
  return Err(Error::WrongAnswer);
}
```

1.2 Question Mark ?

Cách sử dụng “? “trong Xử lý lỗi:

```
foo(30)?;
```

1.2 Custom error

- Import trait **Error**
- Định nghĩa lỗi bằng Struct hoặc Enum
- Implement trait **Error** cho Struct/Enum
- Định nghĩa function bất kì có trả về Result

=> Demo bằng ví dụ

1.2 anyhow and thiserror crate

Có thể sử dụng 1 trong 2 thư viện anyhow hoặc thiserror để xử lý lỗi

```
#[derive(thiserror::Error, Debug)]

pub enum Error {
#[error("Wrong answer")]
WrongAnswer,
#[error("A little bit more")]
More,
#[error("A little bit less")]
Less,
}
```

1.2 Các phương thức của Result

+ Một số phương thức của Result như is_ok, is_err , unwrap ,expect ,...
+ https://doc.rust-lang.org/std/result/enum.Result.html

2. Macro

- Các loại marcos
  - Declarative macros:
    - sử dụng macro_rule!
  - procedural macro: function macros
    - Derive marco : sử dụng cho struct, enum. Ví dụ: Default, Clone, Copy, Debug, PartialEq
    - Attribute-like:
    - Function-like: