5Copy特性使用的局限性

2020年3月9日 12:11

Rust的Copy trait，可以用在存储在栈上的类型。如果一个类型拥有Copy trait，一个旧的变量在赋值给其他变量后仍然可以使用。

局限性：
rust不允许自身或者其任何部分实现了Drop trait的类型使用Copy trait。
1、自身实现了Drop trait。

```
#[derive(Copy, Clone)]
struct MyType {
    a: i32;
}
```

如果不实现Drop trait，是可以编译通过的，但是如果：

```
impl Drop for MyType {
    fn drop(&mut self) {
            .....
    }
}
```

这样是编译通不过的。

2、类型的部分实现了Drop。

```
#[derive(Copy, Clone)]   // 编译时，这里就会报错误了。
struct MyType1;
impl Drop for MyType1 {
    fn drop(&mut self) {..}
}

#[derive(Copy, Clone)]
struct MyType2(MyType1);

#[derive(Copy, Clone)]
struct MyType3{
    a: MyType1;
}

#[derive(Copy, Clone)]
enum MyType4 {
    A(MyType1),
}

#[derive(Copy, Clone)]
enum MyType5 {
    A{a: MyType1},
}
```

以上都是不能编译通过的.