Rust 编程语言入门

杨旭，微软 MVP
Rust、Go 开发者

13. 函数式语言特性:
– 迭代器和闭包

本章内容

- 闭包（closures）
- 迭代器（iterators）
- 优化改善 12 章的实例项目
- 讨论闭包和迭代器的运行时性能

13.1 闭包（1）

- 使用闭包创建抽象行为

什么是闭包（closure）

- 闭包：可以捕获其所在环境的匿名函数。
- 闭包：
  - 是匿名函数
  - 保存为变量、作为参数
  - 可在一个地方创建闭包，然后在另一个上下文中调用闭包来完成运算
  - 可从其定义的作用域捕获值

例子 – 生成自定义运动计划的程序

- 算法的逻辑并不是重点，重点是算法中的计算过程需要几秒钟时间。
- 目标：不让用户发生不必要的等待
  - 仅在必要时调用该算法
  - 只调用一次

- 使用函数重构
- 使用闭包重构

如何定义闭包

- 代码例子：

```
let expensive_closure = |num| {
    println!("calculating slowly...");
    thread::sleep(Duration::from_secs(2));
    num
};
```

例子目前的问题

- 第一个 if 块里，仍然两次调用闭包
- 两个解决思路：
  1. 在 if 块创建本地变量，持有闭包调用的结果
  2. 可使用闭包来解决

再见