MoonBit

现代编程思想

01 课程介绍与程序设计

Zihang Ye, Hongbo Zhang, DII课程组

致谢

本课程参考宾夕法尼亚大学CIS1200课程设计

什么是现代编程思想课

- 这是一门**程序设计**课
  - 课程受众：所有编程爱好者
- 实用技巧
  - 编写较大型程序（~10,000行）
  - 独立分析解决问题
  - 测试驱动开发与设计
- 概念基础
  - 常见数据结构与算法
  - 多种编程范式
  - 关注模块化和组合性

MoonBit 月兔编程

课程工具

- MoonBit月兔
  - 现代静态类型**多范式**编程语言
  - 语法轻量，易上手
  - 浏览器开发环境、云原生开发环境或本地集成开发环境

课程概览

课程	主题	课程	主题
1	课程介绍与程序设计	10	接口：集合与表
2	面向值编程	11	Optional / 结构体 / Unit, Sequencing, 命令
3	函数, 列表与递归	12	可变状态, Aliasing与可变数据结构
4	列表, 元组, 嵌套模式	13	抽象堆栈结构机器
5	数据类型与树	14	可变队列
6	树与二分查找	15	迭代与尾递归
7	二叉搜索树的插入与删除	16	闭包与对象
8	泛型与高阶函数	17	案例：句法分析器与程序解释器
9	高阶函数：Transform与Fold	18	案例：自动积分与小游戏

- 所有课程资料均在互联网上公开
- 课程论坛：taolun.moonbitlang.cn

MoonBit 月兔编程

程序设计

基础设计流程

设计是将非正式的规范转化为可运行代码的过程

推荐采用由测试驱动的开发流程

1. 理解问题
   涉及哪些概念，它们之间存在怎样的联系?
2. 定义接口
   程序应当如何与环境互动?
3. 写测试案例
   对于典型输入，程序应当如何表现？对于非正常输入呢?
4. 实现规定的行为
   经常需要将问题分解为更简单的子问题并对各子问题重复以上流程

一个设计例题

> 超市正在促销，你可以用 `num_exchange` 个空水瓶从超市兑换一瓶水。最开始，你一共购入了 `num_bottles` 瓶水。
> 如果喝掉了水瓶中的水，那么水瓶就会变成空的。
> 给你两个整数 `num_bottles` 和 `num_exchange` ，返回你**最多**可以喝到多少瓶水。
> ——力扣1518

步骤1：理解问题

MoonBit 月兔编程

- 涉及到哪些概念
  - 当前拥有水瓶数量与喝过的水瓶总数
- 它们之间的关系怎样?
  - 初始拥有水瓶数为 `num_bottles`
  - 当拥有的水瓶数超过 `num_exchange` 时，我们可以喝掉 `num_exchange` 瓶水并换得一瓶水，并且继续操作
  - 当拥有的水瓶数少于 `num_exchange` 时，我们只能将拥有的水瓶全部喝完，并结束
- 我们要取得什么?
  - 给定一开始的水瓶数 `num_bottles` 和交换的条件 `num_exchange`，计算最多可以喝多少瓶水

步骤二：定义接口

> 给你两个整数 `num_bottles` 和 `num_exchange` ，返回你**最多**可以喝到多少瓶水。

```
1. fn num_water_bottles(num_bottles: Int, num_exchange: Int) -> Int {
2.     abort("To be done")
3. }
```

步骤三：写测试案例

```
fn init {
    assert(num_water_bottles(9, 3) == 13) // 9 + 3 + 1 = 13
    assert(num_water_bottles(15, 4) == 19)
}

fn assert(test: Bool) {
  if test.not() {
    abort("Test failed")
  }
}
```

步骤四：实现程序

一个可能的实现

```
fn num_water_bottles(num_bottles: Int, num_exchange: Int) -> Int {
  fn consume(num_bottles, num_drunk) {
    if num_bottles >= num_exchange {
      let num_bottles = num_bottles - num_exchange + 1
      let num_drunk = num_drunk + num_exchange
      consume(num_bottles, num_drunk)
    } else {
      num_bottles + num_drunk
    }
  }
  consume(num_bottles, 0)
}

// 省略测试代码
```

MoonBit
月兔编程

运行起来！

MoonBit 月兔编程

小练习

- 案例代码只实现了对于理想的输入的计算。对于非正常输入会出现错误。你能找到这样的非正常输入吗?
    - 提示：在月兔中，Int值域为-2,147,483,648到+2,147,483,647

MoonBit 月兔编程

总结

- 我们希望推动大家采取迭代的设计流程
    1. 理解问题
    2. 定义接口
    3. 定义测试案例
    4. 实现期望的行为
- 现代软件开发依赖**测试驱动开发**
    - 在开发流程中尽早定义测试案例并用它们指导剩余的开发流程