Structure and Interpretation of Computer Programs

(计算机程序的构造与解释（原书第二版）)

参考网址：http://www.mitpress.mit.edu/sicp/

Lisp 语言

运行时调用语言需加入 #lang planet neil/sicp

第一章 构造过程抽象

1.1 程序设计的基本元素

1）基本表达形式
2）组合的方法
3）抽象的方法

1.1.1 表达式

组合式：包含运算对象和前缀运算符，允许嵌套

1.1.2 命名和环境

define 名字与对象关连

全局环境：保持有关的名字-值对偶的轨迹

1.1.3 组合式的求值

树形积累

1.1.4 复合过程

过程定义：一般形式如下：

$$(\text{define}\ (<\text{name}><\text{formal parameters}>)<\text{body}>)$$

1.1.5 过程应用的代换模型

应用序求值：完全展开而后归约，避免对表达式的重复求值（Lisp）

正则序求值：先求值参数而后应用，（解释器）

检测求值方式代码：

```
1. (define (p) (p))
2. (define (test x y)
3.     (if (= x 0)
4.         0
5.         y))
6. (test 0 (p))
```

代码分析：调用 (p) 总是进入一个无限循环(infinite loop)，因为函数 p 会不断调用自身

因此，在应用序中，对 (p) 的求值将使解释器进入无限循环，陷入停滞。在正则序求值中，调用 (p) 从始到终都没有被执行，顺利返回 0。

1.1.6 条件表达式和谓词

分情况分析：cond，一般情况如下

$$(\text{cond } (< \text{p1} > < \text{e1} >) \\ (< \text{p2} > < \text{e2} >) \\ \cdots \\ (< \text{pn} > < \text{en} >))$$

Racket 中用的是[< p1 > < e1 >],每个分支用中括号而不是小括号,并且最后一句判断必须为：[#t　< e >]，即前面的判断都不正确，则进行 e 求值。

Cond 后子句为表达式对偶(<p> <e>)，每个对偶中第一个表达式为谓词。

如果 p 值为 true，则返回相应的序列表达式<e>的值，否则继续往下寻找。

If 表达式的形式　　$(\text{if} < \text{predicate} > < \text{consequent} >< \text{alternative} >)$两个分支只有一个会求值。

三个复合运算符：与 and，或 or，非 not

$$(\text{and} < \text{e1} >< \text{e2} > \cdots < \text{en} >)$$

$$(\text{or} < \text{e1} >< \text{e2} > \cdots < \text{en} >)$$

$$(\text{not} < \text{e} >)$$

and 和 or 是逐步判断，并不是所有子表达式都求值

1.1.7 实例：采用牛顿法求平方根

牛顿逐步逼近：对 x 的平方根的猜测值为 y，新的猜测值尾 y 和 x/y 的平均值。

1.1.8 过程作为黑箱抽象

局部名：约束变量（形式参数的具体名字与过程无关）

内部定义和块结构：过程的形式参数是相对应过程体里的局部参数。

1）块结构：嵌套定义

2）：词法作用域：是形式参数作为内部定义的自由变量

1.2 过程与它们所产生的计算

1.2.1 线性的递归和迭代

递归计算：计算过程构造一个推迟进行的链条，收缩截断表现这些运算的实际执行。

线性递归过程：递归链的长度正比于函数运算数（参数）

迭代计算：固定数目的状态变量描述计算的过程，类似于尾递归。

线性迭代过程：计算步骤随着函数参数线性增长。

1.2.2 树形递归

多参数的递归

1.2.3 增长的阶

记 R(n)为 theta（f(n)）的增长阶，$\mathrm{R(n)} = \theta(\mathrm{f(n)})$

如果存在与 n 无关的整数 k1 和 k2，使得

$$\mathrm{k1\ f(n)} \le \mathrm{R(n)} \le \mathrm{k2\ f(n)}$$

对于任意足够大的 n 都成立

1.2.4 求幂

递归的使用

1.2.5 最大公约数

欧几里得算法：辗转相除

Lame 定理：如果欧几里得算法需要用 k 步计算出一对整数得 GCD，那么这对数中较小得一个数必然大于或则等于第 k 个斐波那契数。

1.2.6 素数检测

1）寻找因子：从 2 开始寻找最小整数因子

2）费马检查：基于费马小定理，概率上的正确性

1.3 用高阶函数做抽象

1.3.1 过程作为参数

类似于柯里化函数

1.3.2 用 lambda 构造过程

与 define 一样的作用，但不为过程提供名字，类似于 sml 中的 fn

let 可以创建局部变量

1.3.3 过程作为一般性的方法

计算过程形式化为一个过程

牛顿法：导数用极限定义，求不动点

抽象和第一级过程：变量命名，提供过程作为参数，由过程作为结果返回，包含在数据结构中