高阶函数

1. 简介

高阶函数（Higher Order Function）是一种以函数为参数的函数。它们都被用于映射、过滤、归档（folding）和排序表。The higher order functions promote modularity of programs.编写对各种情况都适用的高阶函数与为单一情况编写递归函数相比可以使程序更具可读性。比如说，使用一个高阶函数来实现排序可以使得我们使用不同的条件来排序，这就将排序条件和排序过程清楚地划分开来。函数sort具有两个参数，其一是一个待排序的表，其二是定序（Ordering）函数。下面展示了按照大小将一个整数表正序排序。<函数就是（本例中的）两数的定序函数。

(sort '(7883 9099 6729 2828 7754 4179 5340 2644 2958 2239) <)

⇒ (2239 2644 2828 2958 4179 5340 6729 7754 7883 9099)

另一方面，按照每个数末两位的大小排序可以按下面的方式实现：

(sort '(7883 9099 6729 2828 7754 4179 5340 2644 2958 2239)

(lambda (x y) (< (modulo x 100) (modulo y 100))))

⇒ (2828 6729 2239 5340 2644 7754 2958 4179 7883 9099)

正如这里所演示的，像快速排序（Quick Sort）、归并排序（Merge Sort）等排序过程（TBD）

在本节中，我们讲解预定义的高阶函数，然后介绍如何定义高阶函数。由于Scheme并不区别过程和其它的数据结构，因此你可以通过将函数当作参数传递轻松的定义自己的高阶函数。

事实上，TBD，因为Scheme并没有定义块的结构的语法，这是因为lambda表达式就被当作一个块。

1. 映射

映射是将同样的行为应用于表所有元素的过程。R5RS定义了两个映射过程：其一为返回转化后的表的map过程，另一为注重副作用的for-each过程。

* 1. map

map过程的格式如下：

(map **procedure** **list1** **list2** ...)

procedure是个与某个过程或lambda表达式相绑定的符号。作为参数的表的个数视procedure需要的参数而定。

例：

; Adding each item of '(1 2 3) and '(4 5 6).

(map + '(1 2 3) '(4 5 6))

⇒ (5 7 9)

; Squaring each item of '(1 2 3)

(map (lambda (x) (\* x x)) '(1 2 3))

⇒ (1 4 9)

* 1. for-each

for-each的格式与map一致。但for-each并不返回一个具体的值，只是用于副作用。

例：

(define sum 0)

(for-each (lambda (x) (set! sum (+ sum x))) '(1 2 3 4))

sum

⇒ 10

练习1

用map编写下面的函数：

1. 一个将表中所有元素翻倍的函数；
2. 一个将两个表中对应位置元素相减的函数；
3. 过滤

尽管过滤函数并没有在R5RS中定义，但MIT-Scheme实现提供了keep-matching-items和delete-matching-item两个函数。其它实现中应该有类似的函数。

(keep-matching-items '(1 2 -3 -4 5) positive?)

⇒ (1 2 5)

练习2

编写下列函数

1. 滤取（Filtering Out）出一个表中的偶数；
2. 滤取出不满足10 ≤ x ≤ 100的数；
3. 归档

尽管在R5RS中没有定义归档函数，但MIT-Scheme提供了reduce等函数。

(reduce + 0 '(1 2 3 4)) ⇒ 10

(reduce + 0 '(1 2)) ⇒ 3

(reduce + 0 '(1)) ⇒ 1

(reduce + 0 '()) ⇒ 0

(reduce + 0 '(foo)) ⇒ foo

(reduce list '() '(1 2 3 4)) ⇒ (((1 2) 3) 4)

练习3

1. 编写一个将表中所有元素平方的函数，然后求取它们的和，最后求和的平方根。
2. 排序

尽管R5RS中没有定义排序函数，但MIT-Scheme提供了sort（实为merge-sort实现）和quick-sort函数。

(sort '(3 5 1 4 -1) <)

⇒ (-1 1 3 4 5)

练习4

编写下列函数

1、以sin(x)的magnitude升序排序；

2、以表长度降序排序；

1. apply函数

apply函数是将一个过程应用于一个表（译注：将表展开，作为过程的参数）。即使函数有变元个参数，首参数和末参数分别应该是一个过程和一个表。虽然看起来不然，但这个函数的确非常方便。

(apply max '(1 3 2)) ⇒ 3

(apply + 1 2 '(3 4 5)) ⇒ 15

(apply - 100 '(5 12 17)) ⇒ 66

练习5

1. 用apply编写练习3中的函数。
2. 编写高阶函数

自己编写高阶函数非常容易。这里用member-if、member和fractal演示。

7.1 member-if和member

member-if函数使用一个谓词和一个表作为参数，返回一个子表（TBD）