5. 分支

1.简介

在前面的章节，我介绍了如何定义函数。在本章，我会介绍如何通过条件定义过程。这个是编写实际程序的很重要的一步。

2. if表达式

If表达式将过程分为两个部分。If的格式如下：

(**if** **predicate** **then\_value** **else\_value**)

如果predicate部分为真，那么then\_value部分被求值，否则else\_value部分被求值，并且求得的值会返回给if语句的括号外。True是除false以外的任意值，true使用#t表示，false用#f表示。

在R5RS中，false（#f）和空表（’()）是两个不同的对象。然而，在MIT-Scheme中，这两个为同一对象。这个不同可能是历史遗留问题，在以前的标准——R4RS中，#f和’()被定义为同一对象。

因此，从兼容性角度考虑，你不应该使用表目录【待议】作为谓词。使用函数null?来判断表是否为空。

(null? '())

;Value: #t

(null? '(a b c))

;Value: () ;#f

函数not可用于对谓词取反。此函数只有一个参数且如果参数值为#f则返回#t，反之，参数值为#t则返回#f。if表达式是一个特殊形式，因为它不对所有的参数求值。因为如果predicate为真，则只有then\_value部分被求值。另一方面，如果predicate为假，只有else\_value部分被求值。

例：首相为a0，增长率r，项数为n的几何增长（geometric progression）数列之和

(define (sum-gp a0 r n)

(\* a0

(if (= r 1)

n

(/ (- 1 (expt r n)) (- 1 r))))) ; !!

通常来说，几何增长的求和公式如下：

a0 \* (1 - rn) / (1 - r) (r ≠ 1)

a0 \* n (r = 1)

如果if表达式对所有参数求值的话，那么有;!!注释的那行就算在r=1时也会被求值，这将导致产生一个“除数为0”的错误。

你也可以省去else\_value项。这样的话，当predicate为假时，返回值就没有被指定。如果你希望当predicate为假时返回#f，那么就要明确将它写出来。

then\_value和else\_value都应该是S-表达式。如果你需要副作用，那么就应该使用begin表达式。我们将在下一章讨论begin表达式。

练习1

编写下面的函数。阅读第五节了解如何编写谓词。

1. 返回一个实数绝对值的函数。
2. 返回一个实数的倒数的函数。如果参数为0，则返回#f。
3. 将一个整数转化为ASCII码字符的函数。整数可以被转化为33-126号之间的ASCII码。使用integer->char可以将整数转化为字符。如果给定的整数不能够转化为字符，那么就返回#f。

3. and和or

and和or是用于组合条件的两个特殊形式。Scheme中的and和or不同于C语言中的约定。它们不返回一个布尔值（#t或#f），而是返回给定的参数之一。and和or可以使你的代码更加短小。

* 1. and

and具有可变个数的参数，并从左到右对它们求值。如果某一参数为#f，那么它就返回#f，而不对剩余参数求值。反过来说，如果所有的参数都不是#f，那么就返回最后一个参数的值。

(and #f 0)

;Value: ()

(and 1 2 3)

;Value: 3

(and 1 2 3 #f)

;Value: ()

* 1. or

or具有可变个数的参数，并从左到右对它们求值。它返回第一个不是值#f的参数，而余下的参数不会被求值。如果所有的参数的值都是#f的话，则返回最后一个参数的值。

(or #f 0)

;Value: 0

(or 1 2 3)

;Value: 1

(or #f 1 2 3)

;Value: 1

(or #f #f #f)

;Value: ()

练习2

编写下面的函数。

1. 一个接受三个实数作为参数的函数，若参数皆为正数则返回它们的乘积。
2. 一个接受三个实数作为参数的函数，若参数至少一个为负数则返回它们的乘积。

4. cond表达式

尽管所有的分支都可以用if表达式表达，但当条件有更多的可能性时，你就需要使用嵌套的if表达式了，这将使代码变得复杂。处理这种情况可以使用cond表达式。cond表达式的格式如下：

(**cond**

(**predicate\_1** **clauses\_1**)

(**predicate\_2** **clauses\_2**)

......

(**predicate\_n** **clauses\_n**)

(**else** **clauses\_else**))

在cond表达式中，predicates\_i是按照从上到下的顺序求值，而当predicates\_i为真时，clause\_i会被求值并返回。i之后的predicates和clauses不会被求值。如果所有的predicates\_i都是假的话，则返回cluase\_else。在一个子句中，你可以写数条S-表达式，而clause的值是最后一条S-表达式。

例：城市游泳池的收费。

Foo市的城市游泳池按照顾客的年龄收费：

* free if **age** ≤ 3 or **age** ≥ 65.
* 0.5 dollars for 4 ≤ **age** ≤ 6.
* 1.0 dollars for 7 ≤ **age** ≤ 12.
* 1.5 dollars for 13 ≤ **age** ≤ 15.
* 1.8 dollars for 16 ≤ **age** ≤ 18.
* 2.0 dollars for others.

那么，一个返回城市游泳池收费的函数如下：

(define (fee age)

(cond

((or (<= age 3) (>= age 65)) 0)

((<= 4 age 6) 0.5)

((<= 7 age 12) 1.0)

((<= 13 age 15) 1.5)

((<= 16 age 18) 1.8)

(else 2.0)))

练习 3

编写下列函数。

1. 成绩（A-D）是由分数决定的。编写一个将分数映射为成绩的函数，映射规则如下：
   1. A if **score** ≥ 80
   2. B if 60 ≤ **score** ≤ 79
   3. C if 40 ≤ **score** ≤ 59
   4. D if **score** < 40

5. 做出判断的函数

我将介绍一些用于做判断的函数。这些函数的名字都以‘？’结尾。

5.1 eq?、eqv?和equal?

基本函数eq?、eqv?、equal?具有两个参数，用于检查这两个参数是否“一致”。这三个函数之间略微有些区别。

eq?

该函数比较两个对象的地址，如果相同的话就返回#t。例如，(eq? str str)返回#t，因为

str本身的地址是一致的。与此相对的，因为字符串”hello”和”hello”被储存在了不同的地址中，函数将返回#f。不要使用eq?来比较数字，因为不仅在R5RS中，甚至在MIT-Scheme实现中，它都没有指定返回值。使用eqv?或者=替代。

(define str "hello")

;Value: str

(eq? str str)

;Value: #t

(eq? "hello" "hello")

;Value: () ← It should be #f in R5RS

;;; comparing numbers depends on implementations

(eq? 1 1)

;Value: #t

(eq? 1.0 1.0)

;Value: ()

eqv?

该函数比较两个存储在内存中的对象的类型和值。如果类型和值都一致的话就返回#t。对于过程（lambda表达式）的比较依赖于具体的实现。这个函数不能用于类似于表和字符串一类的序列比较，因为尽管这些序列看起来是一致的，但它们的值是存储在不同的地址中。

(eqv? 1.0 1.0)

;Value: #t

(eqv? 1 1.0)

;Value: ()

;;; don't use it to compare sequences

(eqv? (list 1 2 3) (list 1 2 3))

;Value: ()

(eqv? "hello" "hello")

;Value: ()

;;; the following depends on implementations

(eqv? (lambda(x) x) (lambda (x) x))

;Value: ()

equal?

该函数用于比较类似于表或者字符串一类的序列。

(equal? (list 1 2 3) (list 1 2 3))

;Value: #t

(equal? "hello" "hello")

;Value: #t

5.2 用于检查数据类型的函数

下面列举了几个用于检查类型的函数。这些函数都只有一个参数。

pair?

如果对象为

list?

如果对象是一个表则返回#t。要小心的是空表’()是一个表但是不是一个序对。

null?

如果对象是空表’()的话就返回#t。

symbol?

如果对象是一个符号则返回#t。

char?

如果对象是一个字符则返回#t。

string?

如果对象是一个字符串则返回#t。

number?

如果对象是一个数字则返回#t。

complex?

如果对象是一个复数则返回#t。

real?

如果对象是一个实数则返回#t。

rational?

如果对象是一个有理数则返回#t。

integer?

如果对象是一个整数则返回#t。

exact?

如果对象**不是**一个浮点数的话则返回#t。

inexact?

如果对象是一个浮点数的话则返回#t。

5.3 用于比较数的函数

=、>、<、<=、>=

这些函数都有可变个数的参数。如果参数是按照这些函数的名字排序的话，函数就返回#t。

(= 1 1 1.0)

;Value: #t

(< 1 2 3)

;Value: #t

(< 1)

;Value: #t

(<)

;Value: #t

(= 2 2 2)

;Value: #t

(< 2 3 3.1)

;Value: #t

(> 4 1 -0.2)

;Value: #t

(<= 1 1 1.1)

;Value: #t

(>= 2 1 1.0)

;Value: #t

(< 3 4 3.9)

;Value: ()

odd?、even?、postitive?、negative?、zero?

这些函数仅有一个参数，如果这些参数满足函数名所指示的条件话就返回#t。

5.4 用于比较符号的函数

在比较字符的时候可以使用char=?、char<?、char>?、char<=?以及char>=?函数。具体的细节请参见R5RS。

5.5 用于比较字符串的函数

比较字符串时，可以使用string=?和string-ci=?等函数。具体细节请参见R5RS。

6 总结

在这一章中，我总结了关于分支的知识点。编写分支程序可以使用if表达式和cond表达式。

下一章我讲讲解局部变量。

练习答案

**Answer 1**

; 1

(define (my-abs n)

(\* n

(if (positive? n) 1 -1)))

; 2

(define (inv n)

(if (not (zero? n))

(/ n)))

; 3

(define (i2a n)

(if (<= 33 n 126)

(integer->char n)))

**Answer 2**

; 1

(define (pro3and a b c)

(and (positive? a)

(positive? b)

(positive? c)

(\* a b c)))

; 2

(define (pro3or a b c)

(if (or (negative? a)

(negative? b)

(negative? c))

(\* a b c)))

**Answer 3**

(define (score n)

(cond

((>= n 80) 'A)

((<= 60 n 79) 'B)

((<= 40 n 59) 'C)

(else 'D)))