字符与字符串

1. 简介

我只介绍了列表和数，因为它们在Scheme中最为常用。然而，Scheme有像字符（Character）、字符串（String）、符号（Symbol）、向量（Vector）等的其它数据类型，我将在11到14章节中介绍它们。

1. 字符

在某个字符前添加 #\ 来表明该物是一个字符。例如，#\a表示字符a。字符#\Space、#\Tab、#\Linefeed和#\Return分别代表空格（Space）、制表符（Tab），Linefeed和返回（Return）。R5RS中定义了下面的与字符相关的函数。

(char? obj)

如果obj是一个字符则返回#t。

(char=? c1 c3)

如果c1和c2是同一个字符的话则返回#t。

(char->integer c)

将c转化为对应的整数（字符代码，character code）

示例：(char->integer #\a) => 97

(integer->char n)

该函数将一个整数转化为对应的字符。

(char<? c1 c2)

(char<= c1 c2)

(char> c1 c2)

(char>= c1 c2)

这些函数用于比较字符。实际上，这些函数比较的是字符代码的大小。

例如，(char<? c1 c2)等同于(< (char->integer c1) (char->integer c2))

(char-ci=? c1 c2),

(char-ci<? c1 c2),

(char-ci<=? c1 c2),

(char-ci>? c1 c2),

(char-ci>=? c1 c2)

这些比较函数对大小写不敏感。

(char-alphabetic? c),

(char-numeric? c),

(char-whitespace? c),

(char-upper-case? c),

(char-lower-case? c)

这些函数分别用于检测字符c是否为字母、数字、空白符、大写字母或小写字母。

(char-upcase c),

(char-downcase c)

这些函数分别返回字符C对应的大写或小写。

1. 字符串

字符串通过两个闭合的双引号表示。例如，”abc”表示字符串abc。

R5RS定义了下面的函数。

(string? s)

如果s是一个字符则返回#t。

(make-string n c)

返回由n个字符c组成的字符串。The character c can be omitted.

(string-length s)

返回字符串s的长度

(string=? s1 s2)

如果字符串s1和s2相同的话则返回#t。

(string-ref s idx)

返回字符串s中索引为idx的字符（索引从0开始计数）

(string-set! s idx c)

将字符串S中索引为idx的字符设置为c

(substring s start end)

返回字符串s从start开始到end-1处的子串。

(substring “abcdefg” 1 4) => “b c d”

(string-append s1 s2 ...)

连接两个字符串s1和s2

(string->list s)

将字符串s转换为由字符构成的表

(list->string ls)

将一个由字符构成的表转换为字符串

(string-copy s)

复制字符串s

练习1

编写一个函数(title-style)，该函数用于将每个单词的首字母大写。

(title-style "the cathedral and the bazaar")

⇒ "The Cathedral And The Bazaar"

1. 总结

本章讲解了字符和字符串。下章我讲讲解符号。符号是Lisp/Scheme中的一种字符型数据类型。Fast text manipulation is possible using this data type.

答案1

先将字符串转化为表，将空格之后的字符大写，最后将表转换会字符串。

【译注：原文似有误。】

(define (identity x) x)

(define (title-style str)

(let loop ((ls (string->list str))

(w #t)

(acc '()))

(if (null? ls)

(list->string (reverse acc))

(let ((c (car ls)))

(loop (cdr ls)

(char-whitespace? c)

(cons ((if w char-upcase identity) c) acc))))))

;;; Another answer, You can assign caps to the string.

(define (title-style str)

(let ((n (string-length str)))

(let loop ((w #t) (i 0))

(if (= i n)

str

(let ((c (string-ref str i)))

(if w (string-set! str i (char-upcase c)))

(loop (char-whitespace? c) (1+ i)))))))

(title-style "the cathedral and the bazaar")

⇒ "The Cathedral And The Bazaar"