Próximo:Funções anônimas, Anterior:Chamando funções, Acima:Funções [Conteúdo][Índice]

13.6 Funções de Mapeamento

Uma *função de mapeamento* aplica uma determinada função (*não* uma forma especial ou macro) a cada elemento de uma lista ou outra coleção. O Emacs Lisp tem várias dessas funções; esta seção descreve `mapcar`, `mapc`, `mapconcate` `mapcan`, que mapeiam sobre uma lista. Veja Definição de mapatoms , para a função `mapatoms`que mapeia os símbolos em uma obarray. Consulte Definição de maphash , para a função `maphash`que mapeia associações de chave/valor em uma tabela de hash.

Essas funções de mapeamento não permitem tabelas de caracteres porque uma tabela de caracteres é uma matriz esparsa cujo intervalo nominal de índices é muito grande. Para mapear uma tabela de caracteres de uma maneira que lide adequadamente com sua natureza esparsa, use a função `map-char-table`(consulte Tabelas de caracteres).

Função: sequência de *funções* mapcar

`mapcar`aplica a *função* a cada elemento da *sequência* , por sua vez, e retorna uma lista dos resultados.

A sequência de argumentos pode ser qualquer tipo de sequência, exceto uma tabela de caracteres; ou seja, uma lista, um vetor, um vetor bool ou uma string. O resultado é sempre uma lista. O comprimento do resultado é o mesmo que o comprimento da *sequência* . Por exemplo:

```
(mapcar 'car' ((ab) (cd) (ef)))
     ⇒ (ás)
(mapcar '1+ [1 2 3])
     ⇒ (2 3 4)
(mapcar 'string "abc")
     ⇒ ("a" "b" "c")


;; Chame cada função em my-hooks.
(mapcar 'funcall my-hooks)


(defun mapcar* (function &rest args)
  "Aplica FUNCTION a carros sucessivos de todos os ARGS.
Devolva a lista de resultados."
  ;; Se nenhuma lista estiver esgotada,
  (se (não (memq nil args))
      ;; aplicar função aos CARs.
      (cons (função de aplicação (mapcar 'car args)))
            (aplica a função 'mapcar*
                   ;; Recurse para o resto dos elementos.
                   (mapcar 'cdr argumentos)))))


(mapcar* 'contras '(abc) '(1 2 3 4))
     ⇒ ((a . 1) (b . 2) (c . 3))
```

Função: sequência de *funções* mapcan

Esta função aplica *função* a cada elemento de *sequence* , como `mapcar`, mas ao invés de coletar os resultados em uma lista, ela retorna uma única lista com todos os elementos dos resultados (que

devem ser listas), alterando os resultados (usando `nconc`; veja Rearranjo). Como com `mapcar`, a *sequência* pode ser de qualquer tipo, exceto uma tabela de caracteres.

```
;; Contraste isso:
(mapcar 'lista' (abcd))
     ⇒ ((a) (b) (c) (d))
;; com isso:
(mapcan 'lista' (abcd))
     ⇒ (abcd)
```

Função: sequência da *função* mapc

`mapc`é semelhante, `mapcar`exceto que a *função* é usada apenas para efeitos colaterais - os valores que ela retorna são ignorados, não coletados em uma lista. `mapc`sempre retorna *sequência* .

Função: separador de *sequência da função* mapconcat

`mapconcat`aplica *função* a cada elemento da *sequência* ; os resultados, que devem ser sequências de caracteres (strings, vetores ou listas), são concatenados em um único valor de retorno de string. Entre cada par de sequências de resultados, `mapconcat` insere os caracteres do *separador* , que também deve ser uma string, um vetor ou uma lista de caracteres. Veja Sequências Arrays Vetores .

A função argumento deve ser uma função que pode receber um argumento e retornar uma sequência de caracteres: uma string, um vetor ou uma lista. *A sequência* de argumentos pode ser qualquer tipo de sequência, exceto uma tabela de caracteres; ou seja, uma lista, um vetor, um vetor bool ou uma string.

```
(mapconcat 'nome-símbolo
           '(O gato no chapéu)
           " ")
     ⇒ "O gato de chapéu"


(mapconcat (lambda (x) (formato "%c" (1+ x))))
           "HAL-8000"
           "")
     ⇒ "IBM.9111"
```

Próximo:Funções anônimas, Anterior:Chamando funções, Acima:Funções [Conteúdo][Índice]