Próximo:Predicados de igualdade, Anterior:Objetos circulares, Acima:Tipos de dados Lisp [Conteúdo][Índice]

2.7 Tipo de Predicados

O próprio interpretador do Emacs Lisp não realiza verificação de tipo nos argumentos reais passados para funções quando elas são chamadas. Não poderia fazê-lo, pois os argumentos de função em Lisp não possuem tipos de dados declarados, como em outras linguagens de programação. Portanto, cabe à função individual testar se cada argumento real pertence a um tipo que a função pode usar.

Todas as funções internas verificam os tipos de seus argumentos reais quando apropriado e sinalizam um `wrong-type-argument`erro se um argumento for do tipo errado. Por exemplo, aqui está o que acontece se você passar um argumento para `+`o qual ele não pode lidar:

```
(+ 2'a)
     erro→ tipo de argumento errado: número-ou-marcador-p, um
```

Se você quiser que seu programa lide com tipos diferentes de maneira diferente, você deve fazer uma verificação de tipo explícita. A maneira mais comum de verificar o tipo de um objeto é chamar uma função *de predicado de tipo* . O Emacs tem um predicado de tipo para cada tipo, bem como alguns predicados para combinações de tipos.

Uma função de predicado de tipo recebe um argumento; ele retorna `t`se o argumento pertence ao tipo apropriado e `nil`caso contrário. Seguindo uma convenção geral do Lisp para funções de predicado, a maioria dos nomes de predicados de tipo terminam com '`p`'.

Aqui está um exemplo que usa os predicados `listp`para verificar uma lista e `symbolp`um símbolo.

```
(complemento defun (x)
  (cond ((símbolo x)
         ;; Se X for um símbolo, coloque-o em LIST.
         (setq lista (cons x lista)))
        ((listap x)
         ;; Se X for uma lista, adicione seus elementos a LIST.
         (setq lista (anexar x lista)))
        (t
         ;; Lidamos apenas com símbolos e listas.
         (erro "Argumento inválido %s no complemento" x))))
```

Aqui está uma tabela de predicados de tipo predefinidos, em ordem alfabética, com referências a mais informações.

`atom`

Veja átomo .

`arrayp`

Veja arrayp .

`bignump`

Veja float .

bool-vector-p

Veja bool-vector-p .

booleanp

Veja booleanp .

bufferp

Veja buffer .

byte-code-function-p

Veja byte-code-function-p .

case-table-p

Veja case-table-p .

char-or-string-p

Veja char-or-string-p .

char-table-p

Veja char-table-p .

commandp

Veja o comando .

condition-variable-p

Consulte condição-variável-p .

consp

Veja cons .

custom-variable-p

Consulte custom-variable-p .

fixnump

Veja float .

floatp

Veja float .

fontp

Consulte Fonte de baixo nível .

frame-configuration-p

Veja frame-configuration-p .

frame-live-p

Veja frame-live-p .

framep

Veja frame .

functionp

Veja a função .

hash-table-p

Veja tabela de hash-p .

integer-or-marker-p

Consulte integer-or-marker-p .

integerp

Veja inteirop .

keymapp

Veja mapa de teclas .

keywordp

Consulte Variáveis Constantes .

listp

Veja listap .

markerp

Veja marcador .

mutexp

Veja muexp .

nlistp

Veja nlistp .

number-or-marker-p

Consulte número-ou-marcador-p .

numberp

Veja o númerop .

overlayp

Consulte sobreposição .

processp

Veja o processo .

recordp

Veja registro .

sequencep

Veja sequência .

string-or-null-p

Consulte string-or-null-p .

stringp

Veja stringp .

subrp

Veja subpr .

symbolp

Veja simb .

syntax-table-p

Consulte sintaxe-tabela-p .

threadp

Veja thread .

vectorp

Veja vectorp .

wholenump

Veja todonump .

window-configuration-p

Consulte window-configuration-p .

window-live-p

Veja window-live-p .

windowp

Veja janelap .

A maneira mais geral de verificar o tipo de um objeto é chamar a função `type-of`. Lembre-se de que cada objeto pertence a um e apenas um tipo primitivo; `type-of`informa qual (consulte Tipos de dados Lisp). Mas `type-of`não sabe nada sobre tipos não primitivos. Na maioria dos casos, é mais conveniente usar predicados de tipo do que `type-of`.

Função: tipo de *objeto*

Esta função retorna um símbolo nomeando o tipo primitivo de *objeto* . O valor é um dos símbolos `bool-vector`, `buffer`, `char-table`, `compiled-function`, `condition-variable`, `cons`, `finalizer`, `float`, `font-entity`, `font-object`, `font-spec`, `frame`, `hash-table`, `integer`, `marker`, `mutex`, `overlay`, `process`, `string`, `subr`, `symbol`, `thread`, `vector`, `window`, ou `window-configuration`. No entanto, se o *objeto* for um registro, o tipo especificado por seu primeiro slot será retornado; Registros .

```
(tipo de 1)
     ⇒ inteiro
(tipo de 'nil)
     ⇒ símbolo
(tipo de '()) ; ()é nil.
     ⇒ símbolo
```

```
(tipo de '(x))
     ⇒ contras
(tipo de (registro 'foo))
     ⇒ fo
```

Próximo:Predicados de igualdade, Anterior:Objetos circulares, Acima:Tipos de dados Lisp [Conteúdo][Índice]