

Próximo:[Problemas com macros](#), Anterior:[Compilando Macros](#), Acima:[Macros](#) [Conteúdo][Índice]

14.4 Definindo Macros

Um objeto de macro Lisp é uma lista cujo CAR é macroe cujo CDR é uma função. A expansão da macro funciona aplicando a função (com apply) à lista de argumentos não avaliados da chamada da macro .

É possível usar uma macro anônima Lisp como uma função anônima, mas isso nunca é feito, porque não faz sentido passar uma macro anônima para funcionais como mapcar. Na prática, todas as macros Lisp têm nomes e quase sempre são definidas com a defmacro.

Macro: defmacro name args [doc] [declare] body...

defmacro define o nome do símbolo (que não deve ser citado) como uma macro que se parece com isso:

```
(macro lambda args . corpo )
```

(Observe que o CDR desta lista é uma expressão lambda.) Esse objeto de macro é armazenado na célula de função de *name* . O significado de *args* é o mesmo de uma função, e as palavras-chave &reste &optional podem ser usadas (consulte [Argument List](#)). Nem o *nome* nem os *argumentos* devem ser citados. O valor de retorno de defmacro é indefinido.

doc , se presente, deve ser uma string especificando a string de documentação da macro. *declare* , se presente, deve ser um declare formulário especificando metadados para a macro (consulte [Declare Form](#)). Observe que as macros não podem ter declarações interativas, pois não podem ser chamadas interativamente.

As macros geralmente precisam construir grandes estruturas de lista a partir de uma mistura de constantes e partes inconstantes. Para facilitar, use o ``' sintaxe (consulte [Backquote](#)). Por exemplo:

```
(defmacro t-torna-se-nil (variável)
  ` (se (eq ,variável t)
        (setq ,variável nil)))

(t-torna-se-nil foo)
  ≡ (se (eq foo t) (setq foo nil))
```

Próximo:[Problemas com macros](#), Anterior:[Compilando Macros](#), Acima:[Macros](#) [Conteúdo][Índice]