Próximo:Comparação de Números, Anterior:Noções básicas de flutuação, Acima:Números [Conteúdo][Índice]

3.3 Tipo de Predicados para Números

As funções nesta seção testam números ou um tipo específico de número. As funções `integerp`e `floatp`podem receber qualquer tipo de objeto Lisp como argumento (não seriam de muita utilidade de outra forma), mas o `zerop`predicado requer um número como argumento. Veja também `integer-or-marker-p`e `number-or-marker-p`, em Predicados sobre Marcadores .

Função: *objeto* bignump

Esse predicado testa se seu argumento é um inteiro grande e retorna em `t`caso afirmativo, `nil`caso contrário. Ao contrário dos pequenos inteiros, os grandes inteiros podem ser `=`ou `eql`mesmo se não forem `eq`.

Função: *objeto* fixnump

Esse predicado testa se seu argumento é um inteiro pequeno e retorna em `t`caso afirmativo, `nil`caso contrário. Pequenos inteiros podem ser comparados com `eq`.

Função: *objeto* floatp

Esse predicado testa se seu argumento é de ponto flutuante e retorna em `t`caso afirmativo, `nil`caso contrário.

Função: *objeto* integerp

Esse predicado testa se seu argumento é um inteiro e retorna `t`se for, `nil`caso contrário.

Função: *objeto* numberp

Esse predicado testa se seu argumento é um número (inteiro ou ponto flutuante) e retorna em `t`caso afirmativo, `nil`caso contrário.

Função: *objeto* natnump

Esse predicado (cujo nome vem da frase “número natural”) testa para ver se seu argumento é um inteiro não negativo e retorna `t`se for, `nil`caso contrário. 0 é considerado não negativo.

`wholenump`é sinônimo de `natnump`.

Função: *número* zerop

Esse predicado testa se seu argumento é zero e retorna em `t` caso afirmativo, `nil`caso contrário. O argumento deve ser um número.

`(zerop x)`é equivalente a `(= x 0)`.

Próximo:Comparação de Números, Anterior:Noções básicas de flutuação, Acima:Números [Conteúdo][Índice]