Informação Simbólica

Fundamentos de Algoritmos

INF05008

Nomeando Valores

- Quando um **valor ocorre muitas vezes** em um programa, é interessante **atribuir-lhe um nome**
- Exemplo: o valor de π é nomeado da seguinte forma:

```
(define PI 3.14)

(define (área-do-disco r)
    (* PI (* r r)))
```

- Para usar uma aproximação melhor de π, não precisamos buscar todas as ocorrências, **basta mudarmos o valor associado a ele na definição**

Informação Simbólica

- Computadores processam informações **simbólicas** tais como nomes, palavras, direções, imagens, etc.

- **Linguagens de programação** têm suporte para pelo menos uma **forma de representar informação simbólica**

- Scheme tem suporte para
  - Símbolos
  - Strings
  - Caracteres (do teclado)
  - Imagens

Símbolos

- Um **símbolo** é uma **sequência de caracteres** do teclado precedida por um apóstrofe
- Exemplos:

```
'Ana    'a    'gato!    'dois^3    'entre%outros?
```

- **Interpretação** do símbolo depende do **usuário** e do **contexto**
- Assim como os números, símbolos são dados **atômicos**
- Somente uma **operação básica** sobre símbolos: `symbol=?`

Símbolos (cont.)

- Exemplos de expressões com `symbol=?`:
- `(symbol=? 'Hello 'Hello)` é **verdadeiro**
- `(symbol=? 'Hello 'Howdy)` é **falso**
- `(symbol=? 'Hello x)` é **verdadeiro** se x for `'Hello` e **falso**, caso x seja qualquer **outro símbolo**

Usando Símbolos

Função 'resposta' que responde com um comentário às seguintes saudações: "Bom dia", "Tudo bem?", "Boa tarde" e "Boa noite". Cada uma dessas saudações pode ser representada como um símbolo: `'BomDia`, `'TudoBem?`, `'BoaTarde` e `'BoaNoite`.

```
;; resposta : símbolo  ->  símbolo
;; Determina resposta para uma saudação

(define (resposta s) ...)
```

Usando Símbolos (cont.)

A função deve distinguir entre quatro situações. De acordo com a estrutura de projeto, temos uma expressão condicional com quatro cláusulas:

```
(define (resposta s)
 (cond
   [(symbol=? s 'BomDia)    ...]
   [(symbol=? s 'TudoBem?)     ...]
   [(symbol=? s 'BoaTarde) ...]
   [(symbol=? s 'BoaNoite)    ...]))
```

Usando Símbolos (cont.)

A partir do *template*, a função final é facilmente definida. Eis uma possibilidade:

```
(define (resposta s)
    (cond
        [(symbol=? s 'BomDia)    'Oi]
        [(symbol=? s 'TudoBem?)     'Tranquilo]
        [(symbol=? s 'BoaTarde) 'PrecisoDormir]
        [(symbol=? s 'BoaNoite)   'EstouMuitoCansado]))
```

Strings

- **Strings** são sequências de caracteres do teclado entre aspas duplas
- Exemplos:

```
"o cão"       "pois é"      "talvez"
"chocolate"     "dois^3"     "e etc."
```

- **Operação básica**: `string=?` compara duas *strings*
- *Strings* **não são atômicas** (mais sobre isso em listas. . .)

Caracteres

- **Caracteres** também são representados em Scheme

```
#\a, #\b ... #\z

#\A, #\B ... #\Z

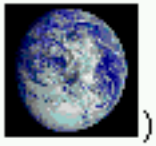
#\0, #\1 ... #\9

#\+, #\-, #\$, #\!, #\space, ...
```

- **Operação básica**: `char=?` compara caracteres

Imagens

- DrScheme também aceita **imagens** como informação simbólica
  - Imagens são **valores**, assim como números e booleanos
  - Podem ser usadas **em qualquer expressão**
  - Para facilitar a referência, é usual **nomeá-las**
  - Pode-se usar a **operação** `image=?` para **comparar imagens**

```
(define EARTH  )

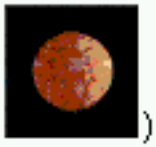
(define MARS  )
```

figura 7, página 47 de HTDP e exercício 5.5.1

Qual é o resultado de cada expressão?

```
(define a 2)
(define b 4)

(define Castelo [Castelo-img])
(define Palácio [Castelo-img])

(+ #\a a)

(+ #\a #\b)

(+ a (sqr b))
```

Qual é o resultado de cada expressão?

```
(define a 2)
(define b 4)

(define Castelo [Castelo-img])
(define Palácio [Castelo-img])

(+ #\a a) = ERRO! Não é possível somar um caracter a um número

(+ #\a #\b)

(+ a (sqr b))
```

Qual é o resultado de cada expressão?

```
(define a 2)
(define b 4)

(define Castelo [Castelo-img])
(define Palácio [Castelo-img])

(+ #\a a) = ERRO! Não é possível somar um caracter a um número

(+ #\a #\b) = ERRO! Operação aplicável apenas a números

(+ a (sqr b))
```

Qual é o resultado de cada expressão?

```
(define a 2)
(define b 4)

(define Castelo [Castelo-img])
(define Palácio [Castelo-img])

(+ #\a a) = ERRO! Não é possível somar um caracter a um número

(+ #\a #\b) = ERRO! Operação aplicável apenas a números

(+ a (sqr b)) = 18
```

```
(char=? #\a #\b)

(char=? #\aa #\a)

(symbol=? #\a 'a)

(symbol=? '#\a 'a)

(symbol=? '\a 'a)

(string=? "aa" "a a")

(string=? "#\a" "#\a")

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a)

(symbol=? #\a 'a)

(symbol=? '#\a 'a)

(symbol=? '\a 'a)

(string=? "aa" "a a")

(string=? "#\a" "#\a")

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a) = ERRO! Caracter incorreto: #\aa

(symbol=? #\a 'a)

(symbol=? '#\a 'a)

(symbol=? '\a 'a)

(string=? "aa" "a a")

(string=? "#\a" "#\a")

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a) = ERRO! Caracter incorreto: #\aa

(symbol=? #\a 'a) = ERRO! Operação válida somente para símbolo

(symbol=? '#\a 'a)

(symbol=? '\a 'a)

(string=? "aa" "a a")

(string=? "#\a" "#\a")

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a) = ERRO! Caracter incorreto: #\aa

(symbol=? #\a 'a) = ERRO! Operação válida somente para símbolo

(symbol=? '#\a 'a) = ERRO! Símbolo incorreto: '#\a

(symbol=? '\a 'a)

(string=? "aa" "a a")

(string=? "#\a" "#\a")

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a) = ERRO! Caracter incorreto: #\aa

(symbol=? #\a 'a) = ERRO! Operação válida somente para símbolo

(symbol=? '#\a 'a) = ERRO! Símbolo incorreto: '#\a

(symbol=? '\a 'a) = true

(string=? "aa" "a a")

(string=? "#\a" "#\a")

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a) = ERRO! Caracter incorreto: #\aa

(symbol=? #\a 'a) = ERRO! Operação válida somente para símbolo

(symbol=? '#\a 'a) = ERRO! Símbolo incorreto: '#\a

(symbol=? '\a 'a) = true

(string=? "aa" "a a") = false

(string=? "#\a" "#\a")

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a) = ERRO! Caracter incorreto: #\aa

(symbol=? #\a 'a) = ERRO! Operação válida somente para símbolo

(symbol=? '#\a 'a) = ERRO! Símbolo incorreto: '#\a

(symbol=? '\a 'a) = true

(string=? "aa" "a a") = false

(string=? "#\a" "#\a") = true

(string=? "Hi" "hi")
```

```
(char=? #\a #\b) = false

(char=? #\aa #\a) = ERRO! Caracter incorreto: #\aa

(symbol=? #\a 'a) = ERRO! Operação válida somente para símbolo

(symbol=? '#\a 'a) = ERRO! Símbolo incorreto: '#\a

(symbol=? '\a 'a) = true

(string=? "aa" "a a") = false

(string=? "#\a" "#\a") = true

(string=? "Hi" "hi") = false
```

```
(image=? Castelo "Castelo")

(image=? Castelo Palácio)

(+ Castelo a)

(or (symbol=? 'a 'b) (= a 2))
```

```
(image=? Castelo "Castelo") = ERRO! Operação para imagens

(image=? Castelo Palácio)

(+ Castelo a)

(or (symbol=? 'a 'b) (= a 2))
```

```
(image=? Castelo "Castelo") = ERRO! Operação para imagens

(image=? Castelo Palácio) = true

(+ Castelo a)

(or (symbol=? 'a 'b) (= a 2))
```

```
(image=? Castelo "Castelo") = ERRO! Operação para imagens

(image=? Castelo Palácio) = true

(+ Castelo a) = ERRO! Operação válida apenas para números

(or (symbol=? 'a 'b) (= a 2))
```

```
(image=? Castelo "Castelo") = ERRO! Operação para imagens

(image=? Castelo Palácio) = true

(+ Castelo a) = ERRO! Operação válida apenas para números

(or (symbol=? 'a 'b) (= a 2)) = true
```

1. Uma loja dá descontos nas suas mercadorias dependendo do tempo em que elas estão no estoque. Se um item não foi vendido em 2 semanas, o seu preço cai em 25%. Na terceira semana, o preço cai em 50%, e na quarta, em 75%. A partir da quinta semana, não são dados mais descontos. Faça uma função que, dados o preço de uma mercadoria e o número de semanas em que ela está no estoque, calcula o seu novo valor de venda.

2. Desenvolva a função `testa-chute` que consome dois números, chamados `chute` e `valor` e, dependendo da relação entre eles, devolve uma das seguintes opções: `'MuitoBaixo`, `'Exato!`, `'MuitoAlto`.

3. Desenvolva a função `testa-cores`, a qual implementa um jogo de adivinhação de cores. Esta função recebe as cores atribuídas a duas posições `p1` e `p2` e duas cores de adivinhação `c1` e `c2`. As respostas possíveis são: `'Exato!`, se as cores de adivinhação corresponderem às cores atribuídas às posições na mesma ordem; `'UmaPosiçãoCorreta`, se apenas uma cor de adivinhação corresponder à cor e posição corretas; `'UmaCorCorreta`, se pelo menos uma das cores fornecidas como adivinhação corresponder a uma das cores atribuídas a uma das posições, mas na posição incorreta; e `'TudoErrado!`, caso nenhuma das outras respostas se aplique.

```
;; valor-de-venda : número número -> número
;; Calcula o valor de venda de um produto
;; dados o valor atual do mesmo e o número
;; de semanas em que este está no estoque

(define (valor-de-venda preço-produto nro-semanas)
  (cond
    [(<= nro-semanas 2) (- preço-produto (* 0.25 preço-produto))]
    [(= nro-semanas 3) (- preço-produto (* 0.50 preço-produto))]
    [(= nro-semanas 4) (- preço-produto (* 0.75 preço-produto))]
    [(>= nro-semanas 5) preço-produto]))

;; Exemplos
(valor-de-venda 100 0) ;; produz 75
(valor-de-venda 100 2) ;; produz 75
(valor-de-venda 100 3) ;; produz 50
(valor-de-venda 100 4) ;; produz 25
(valor-de-venda 100 5) ;; produz 100
```

```
;; testa-chute : número número -> símbolo
;; Verifica a relação do número 'chute' com
;; um número 'valor' e retorna uma
;; mensagem correspondente

(define (testa-chute chute valor)
  (cond
    [(< chute valor) 'MuitoBaixo]
    [(> chute valor) 'MuitoAlto]
    [(= chute valor) 'Exato!]))

;; Exemplos
(testa-chute 2 5) ;; produz 'MuitoBaixo
(testa-chute 7 5) ;; produz 'MuitoAlto
(testa-chute 5 5) ;; produz 'Exato!
```

```
;; testa-cores : símbolo símbolo símbolo símbolo -> símbolo
;; Verifica se as cores atribuídas às posições 'p1' e 'p2'
;; correspondem às adivinhações 'c1' e 'c2', na mesma
;; ordem

(define (testa-cores p1 p2 c1 c2)
  (cond
    [(and (symbol=? p1 c1) (symbol=? p2 c2)) 'Exato!]
    [(or (symbol=? p1 c1) (symbol=? p2 c2)) 'UmaPosiçãoCorreta]
    [(or (symbol=? p1 c2) (symbol=? p2 c1)) 'UmaCorCorreta]
    [else 'TudoErrado!]))

;; Exemplos
(testa-cores 'vermelho 'azul 'vermelho 'azul) ;; produz 'Exato!
(testa-cores 'vermelho 'azul 'vermelho 'amarelo) ;; produz
'UmaPosiçãoCorreta
(testa-cores 'vermelho 'azul 'rosa 'azul) ;; produz 'UmaPosiçãoCorreta
(testa-cores 'vermelho 'azul 'azul 'preto) ;; produz 'UmaCorCorreta
```

```
(testa-cores 'vermelho 'azul 'marrom 'vermelho) ;; produz
'UmaCorCorreta
(testa-cores 'vermelho 'azul 'azul 'vermelho) ;; produz 'UmaCorCorreta
(testa-cores 'vermelho 'azul 'preto 'amarelo) ;; produz 'TudoErrado!
```