Cap. 07 - Tipos Mistos

INF05008 - Fundamentos de Algoritmos

Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brasil
http://www.inf.ufrgs.br

Vários Tipos de Dados

- Até aqui, as funções que definimos usavam 4 tipos de dados:
  - elementos do conjunto `Número` : representando **informações numéricas**;
  - elementos do conjunto `Booleano` : representando **valores-verdade**;
  - elementos do conjunto `Número` : representando **informação simbólica**;
  - `struct`: representando **composição de informações**.
- As vezes, precisamos definir funções que processam uma **classe de dados**, incluindo números e estruturas ou estruturas de vários tipos diferentes, por exemplo.
- Como podemos **definir** essas funções e também **proteger** nossas funções de usos indevidos?

Misturando e Distinguindo Dados

- **Problema:** Uma definição alternativa de posições em um plano usa, em vez de um par de coordenadas (x, y), apenas a coordenada x caso o ponto esteja no eixo x (ou seja, quando $y = 0$). Reconstrua a função `distância-para-0` que calcula a distância de um ponto até o 0, agora levando em consideração que a entrada pode ser do tipo `Posn` ou do tipo `Numero`.
  - se for um elemento de *Posn*:
    `(distância-para-0 (make-posn 3 4))` produz 5
  - se for um elemento de *Número*:
    `(distância-para-0 7)` produz 7

- Como descobrir o **tipo da entrada**, já que esta pode ser uma estrutura (no caso, `posn`) ou um número?

Operações para Distinguir Dados

- Racket oferece operações predefinidas para identificar tipos de dados:
  - `number?`: Retorna `true` se o valor ao qual a função é aplicada é do tipo **Número** e `falso`, caso contrário;
  - `boolean?`: Retorna `true` se o valor ao qual a função é aplicada é do tipo **Booleano** (um valor-verdade) e `falso`, caso contrário;
  - `symbol?`: Retorna `true` se o valor ao qual a função é aplicada é do tipo **Símbolo** e `falso`, caso contrário;
  - `struct?`: Retorna `true` se o valor ao qual a função é aplicada é uma **estrutura** e `falso`, caso contrário.

Operações para Distinguir Dados

- Além disso, para cada definição de estrutura são gerados, **automaticamente**, operações para identificar valores dessas novas classes. Por exemplo, para as definições a seguir

```
(define-struct posn (x y))

(define-struct astro (sobrenome nome instrumento vendas))

(define-struct avião (tipo veloc-máx capacidade preço))
```

  são gerados as seguintes operações:

  - `posn?`
  - `astro?`
  - `avião?`

Exercícios

- Avalie as seguintes expressões:

1. `(number? (make-posn 2 3))`
2. `(number? (+ 12 10))`
3. `(posn? 23)`
4. `(posn? (make-posn 23 3))`
5. `(astro? (make-posn 23 3))`

Função `distância-para-0`

- **Definição de dados:**
  Um elemento do conjunto `Pixel-2` é
  1. ou um elemento de Número (`Number`),
  2. ou um elemento de `Posn`.
- **Contrato, objetivo e cabeçalho:**

```
;; distância-para-0 : Pixel-2  ->  Número

;; Calcular a distância de um ponto (um-pixel)
;; para a origem

(define (distância-para-0 um-pixel) ...)
```

Função distância-para-0

- **Exemplos:**

```
 ;; caso a função receba um número:
(check-expect (distância-para-0 7) 7)

;; caso a função receba um posn :
(check-expect
  (distância-para-0 (make-posn 3 4 ) ) 5)
```

Função `distância-para-0`

- Vamos definir a função por etapas...

- **Etapa 1**: Como identificar o **tipo da entrada**?

Função `distância-para-0`

- **Etapa 1**: Como identificar o **tipo da entrada**?

- Usando uma expressão `cond`:

```
(define (distância-para-0 um-pixel)
  (cond
    [(number? um-pixel) ...]
    [(posn? um-pixel) ...]))
```

Função `distância-para-0`

- **Etapa 2**: Como **selecionar as coordenadas**, caso a entrada seja uma estrutura `posn`?

Função `distância-para-0`

- **Etapa 2**: Como **selecionar as coordenadas**, caso a entrada seja uma estrutura `posn`?

- Usando as funções seletoras `posn-x` e `posn-y`:

```
(define (distância-para-0 um-pixel)
  (cond
    [(number? um-pixel) ...]
    [(posn? um-pixel)...(posn-x um-pixel)...
                        (posn-y um-pixel)...]))
```

Função `distância-para-0`

- **Etapa 3**: Como **calcular a função** em cada caso?

Função distância-para-0

- **Etapa 3**: Como **calcular a função** em cada caso?

```
(define (distância-para-0 um-pixel)
  (cond
    [(number? um-pixel) um-pixel]
    [(posn? um-pixel)
      (sqrt (+ (sqr (posn-x um-pixel))
               (sqr (posn-y um-pixel))))]))
```

Desenvolvendo Funções com Dados Mistos

- Fases do projeto de algoritmos a serem modificadas:
  - **Análise e projeto de dados**: Determinar classes distintas de dados. Definição de dados terá cláusulas enumerando os tipos de dados do problema;
  - **Exemplos**: Fornecer um exemplo para cada caso;
  - **Definição da função**: Usando o condicional, o problema é dividido em vários subproblemas. Cada cláusula de `cond` é tratada separadamente.

- As outras fases (**Contrato**, **Objetivo**, **Cabeçalho**, e **Testes**) não sofrem modificações pelo uso de dados com tipos mistos.

Projeto de Algoritmos usando Dados Mistos

Fase	Objetivo	Atividade
Projeto e Análise de Dados	Formular uma definição de dados	Determinar **quantas classes distintas de objetos** tem o problema, **enumerar as alternativas em uma definição de dados**, fazer as definições de estruturas que forem necessárias
Contrato, Objetivo e Cabeçalho	Dar um nome à função, especificar as classes de entrada e saída, descrever o objetivo e formular um cabeçalho	Nomear a função, as classes de entrada e saída e especificar um objetivo

Projeto de Algoritmos usando Dados Mistos

Fase	Objetivo	Atividade
Exemplos	Caracterizar a relação entrada-saída através de exemplos	Criar exemplos da relação entrada-saída, levando em consideração que deve existir pelo menos um exemplo para cada subclasse de dados a qual a função pode ser aplicada
Template	Formular um esboço para a função	Criar um esqueleto da operação `cond` com uma linha para cada tipo de dado. Identificar cada um dos tipos usando operações de identificação de tipos

Projeto de Algoritmos usando Dados Mistos

Fase	Objetivo	Atividade
Corpo	Completar a definição da função	Construir expressões Racket para cada uma das cláusulas do `cond`
Testes	Encontrar erros	Aplicar a função aos exemplos e verificar se os resultados são os esperados

Função que calcula o perímetro de uma forma

```
;; Definição de dados:

(define-struct círculo (centro raio))

;; Um elemento círculo de Círculo é uma estrutura
;; (make-círculo um-centro um-raio) onde
;; um-centro : Posn é um ponto
;; um-raio : Número é um raio
```

Função que calcula o perímetro de uma forma (cont.)

```
;; Definição de dados:

(define-struct quadrado (ponto lado))

;; Um elemento quadrado de Quadrado é uma
;; estrutura
;; (make-quadrado um-ponto um-lado) onde
;; um-ponto : Posn é um ponto
;; um-lado : Número é um comprimento
```

Função que calcula o perímetro de uma forma (cont.)

```
;; Definição de dados:

;; Um elemento forma de Forma é:
;; 1) um elemento de Círculo
;; ou
;; 2) um elemento de Quadrado
```

Função que calcula o perímetro de uma forma (cont.)

```
;; perímetro : Forma  ->  Número
;; Computar o perímetro de uma forma

(define (perímetro uma-forma) ... )

;; Exemplos:

(check-expect
 (perímetro (make-quadrado (make-posn 0 0) 3)) 12)

(check-expect
 (perímetro (make-círculo (make-posn 0 0) 1)) (* 2 PI))
```

- obs.: na realidade, o centro do círculo e o ponto do quadrado não são relevantes para o cálculo do perímetro

Função que calcula o perímetro de uma forma (cont.)

```
;; Template da função:

;; (define (perímetro uma-forma)
;;   (cond
;;     [(quadrado? uma-forma)
;;        ...(quadrado-ponto uma-forma)...
;;        ...(quadrado-lado uma-forma)...]
;;     [(círculo? uma-forma)
;;        ...(círculo-centro uma-forma)...
;;        ...(círculo-raio uma-forma)...]))
```

Função que calcula o perímetro de uma forma (cont.)

```
;; Definição da função:

(define (perímetro uma-forma)
  (cond
    [(círculo? uma-forma)
    (* 2  PI (círculo-raio uma-forma) )]
    [(quadrado? uma-forma)
    (* (quadrado-lado uma-forma) 4)]))
```

Exercício

1. Desenvolva estruturas e definições de dados para o tipo de dados `Forma`, que representa objetos 3D. A coleção deve incluir:
   - elementos do conjunto dos Cubos: o atributo relevante é o tamanho do lado;
   - elementos do conjunto de Prismas: sólidos retangulares cujos atributos relevantes são a altura, a largura e a profundidade;
   - elementos do conjunto de Esferas: o atributo importante é o raio.
2. Desenvolva a função `volume`, que consome uma forma e produz o volume da forma. Dica : o volume de uma esfera de raio r é $\frac{4}{3} * \pi * r^3$.
3. Desenvolva a função `mesma-forma?` que consome duas formas tridimensionais e retorna `true` somente se ambas possuírem **exatamente** as mesmas dimensões.

Soluções

```
(define PI 3.14)

(define-struct cubo (lado))

;; Um elemento cubo de Cubo é uma estrutura
;; (make-cubo lado) onde
;; lado : Número é um lado do cubo

(define-struct prisma (altura largura profundidade))

;; Um elemento prisma de Prisma é uma estrutura
;; (make-prisma altura largura profundidade) onde
;; altura : Número é a altura do prisma
;; largura : Número é a largura do prisma
;; profundidade : Número é a profundidade do prisma
```

Soluções (cont.)

```
(define-struct esfera (raio))

;; Uma elemento esfera de Esfera é uma estrutura
;; (make-esfera raio) onde
;; raio : Número é o raio da esfera

;; Um elemento forma de Forma pode ser
;; 1) um elemento de Cubo    ou
;; 2) um elemento de Prisma  ou
;; 3) um elemento de Esfera
```

Soluções (cont.)

```
; exemplos (uso da função volume):

; (check-expect (volume (make-esfera 1)) (* (/ 4 3) PI))

; (check-expect (volume (make-cubo 1)) 1)

; (check-expect (volume (make-prisma 1 1 1)) 1)
```

Soluções (cont.)

```
;; volume : Forma -> Número
;; Calcula o volume de uma forma tridimensional.
;; Assume-se que a base de um prisma é um retângulo

(define (volume uma-forma)
  (cond
    [(cubo? ... cubo-lado ...
    [(prisma?
     ... prisma-largura ...
     ... prisma-profundidade ...
     ... prisma-altura ...
    [(esfera? ... esfera-raio ...
```

Soluções (cont.)

```
;; volume : Forma -> Número
;; Calcula o volume de uma forma tridimensional.
;; Assume-se que a base de um prisma é um retângulo

(define (volume uma-forma)
  (cond
    [(cubo? uma-forma) (expt (cubo-lado uma-forma) 3)]
    [(prisma? uma-forma)
    (* (prisma-largura uma-forma)
       (prisma-profundidade uma-forma)
       (prisma-altura uma-forma))]
    [(esfera? uma-forma)
    (* (/ 4 3) PI
       (expt (esfera-raio uma-forma) 3))]))
```

Soluções (cont.)

```
 ;; exemplos (uso da função mesma-forma?):

(check-expect (mesma-forma? (make-cubo 1) (make-cubo 1)) true)
(check-expect (mesma-forma? (make-cubo 1) (make-cubo 2)) false)

(check-expect (mesma-forma? (make-esfera 1) (make-esfera 1)) true)
; forneça exemplo para esferas diferentes !

(check-expect (mesma-forma? (make-prisma 1 1 1)
                            (make-prisma 1 1 1)) true)
; forneça exemplo para prismas diferentes !

(check-expect (mesma-forma? (make-cubo 1) (make-esfera 1)) false)
```

Soluções (cont.)

```
;; mesma-forma? : Forma Forma -> Boolean
;; Identifica formas iguais que possuem as
;; mesmas dimensões

(define (mesma-forma? forma1 forma2)
  (cond
 ; caso ambas formas sejam cubos:
    [(and  (cubo? forma1) (cubo? forma2) ...
 ; caso ambas formas sejam prismas:
    [(and (prisma? forma1) (prisma? forma2)...
 ; caso ambas formas sejam esferas:
    [(and (esfera? forma1) (esfera? forma2) ...
    [else ....
```

Soluções (cont.)

```
;; mesma-forma? : Forma Forma -> Boolean
;; Identifica formas iguais que possuem as
;; mesmas dimensões

(define (mesma-forma? forma1 forma2)
  (cond
    [(and (cubo? forma1) (cubo? forma2))
     (= (cubo-lado forma1) (cubo-lado forma2))]
    [(and (prisma? forma1) (prisma? forma2))
     (and (= (prisma-largura forma1)
             (prisma-largura forma2))
          (= (prisma-profundidade forma1)
             (prisma-profundidade forma2))
          (= (prisma-altura forma1)
             (prisma-altura forma2)))]
    [(and (esfera? forma1) (esfera? forma2))
     (= (esfera-raio forma1) (esfera-raio forma2))]
    [else false]))
```

Soluções (cont.)

- Esta é a versão com `or` (vamos preferir esta !!!)

```
  ;; mesma-forma? : Forma Forma -> Boolean
  ;; Identifica formas iguais que possuem as
  ;; mesmas dimensões

(define (mesma-forma? forma1 forma2)
  (or
   (and ; se as 3 condições abaixo forem true, retorna true
   (cubo? forma1) (cubo? forma2) (= (cubo-lado forma1)(cubo-lado forma2))
  ; senão testa para prisma:
  (and (prisma? forma1) (prisma? forma2)
       (= (prisma-largura forma1) (prisma-largura forma2) )
       (= (prisma-profundidade forma1) (prisma-profundidade forma2))
       (= (prisma-altura forma1) (prisma-altura forma2)))
  ; e por fim, para esfera
  (and (esfera? forma1) (esfera? forma2)
       (= (esfera-raio forma1) (esfera-raio forma2))))
  ; se nenhuma das 3 for true, false será a resposta
)
```

Revisitando Exercícios Passados

1. Crie uma função que, dadas informações de um aluno (nome, turma, nível e professor) e o nome de um professor, caso este professor seja o professor do aluno em questão, retorne este professor; caso contrário, retorne a mensagem "Este professor não é professor deste aluno". Considere que as informações relevantes sobre professores são: nome, turma e nível, e que nomes são strings, turmas são números, e níveis são símbolos.

2. Desenvolva a função *médias* que consome 4 números e realiza cálculo de médias. A função deve fazer o cálculo correto dependendo do valor do primeiro número informado: caso ele seja 1, a saída deve ser o média aritmética dos outros três valores; se for 2, o cálculo dever ser da média harmônica. Caso o primeiro número informado seja outro, retornar "Que tipo de média você deseja?"

Revisitando Exercícios Passados

3 Desenvolva uma função que consuma duas `datas` do mesmo ano e produza como resposta quantos dias se passaram da primeira data para a segunda. Assuma que a segunda data é posterior à primeira. Se as duas datas forem de meses diferentes, retornar "Forneça duas datas no mesmo mês!".

4 Desenvolva uma função que calcule a diferença de meses entre uma data `data1` e uma data `data2` do mesmo ano, sendo que estas são fornecidas conforme a estrutura definida anteriormente. Caso as duas datas não sejam do mesmo ano, a função deve retornar a mensagem "As datas devem ser do mesmo ano."