Vários Tipos de Dados

Até aqui, as funções que definimos usavam 4 tipos de dados:

- `number`: representando informações numéricas;
- `boolean`: representando valores-verdade;
- `symbol`: representando informação simbólica;
- `structure`: representando composição de informações.

Às 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, ao invés de um par de coordenadas (x, y) , apenas a coordenada x caso o ponto esteja no eixo x (ou seja, quando $y = 0$). Construa uma função que calcula a distância de um ponto até o 0, levando em consideração que a entrada pode ser ou do tipo (x, y) , ou x .

Como descobrir o tipo da entrada, que pode ser uma estrutura `posn x y` ou um `number`?

Predicados (funções booleanas) para Distinguir Dados

Scheme oferece predicados pré-definidos para identificar tipos de dados:

- `number?`: retorna `true` se o valor ao qual a função é aplicado é um **número**, e `false` caso contrário;
- `boolean?`: retorna `true` se o valor ao qual a função é aplicado é um **valor-verdade**, e `false` caso contrário;
- `symbol?`: retorna `true` se o valor ao qual a função é aplicado é um **símbolo**, e `false` caso contrário;
- `structure?`: retorna `true` se o valor ao qual a função é aplicado é uma **estrutura**, e `false` caso contrário.

Predicados para Distinguir Dados

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

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

```
(define-struct star (last first dob ssn))
```

```
(define-struct airplane (kind max-speed max-load price))
```

são gerados os seguintes predicados

- `posn?`, consome um valor e retorna verdadeiro se esse valor é uma estrutura do tipo `posn`, e falso, caso contrário;
- `star?`, ...
- `airplane?`, ...

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. `(star? (make-posn 23 3))`

Função `distance-to-0`

Definição de dados: Um `pixel-2` é ou

1. um número (`number`), ou
2. uma estrutura `posn`.

Contrato, objetivo e cabeçalho:

```
;; distance-to-0 : pixel-2 -> number
;; obj: calcular a distancia de um ponto (a-pixel)
;;      para a origem
(define (distance-to-0 a-pixel) ...)
```

Função `distance-to-0`

Vamos definir a função por etapas.

Etapa 1: Como identificar o tipo da entrada? Usando uma expressão `cond`:

```
(define (distance-to-0 a-pixel)
  (cond
    [(number? a-pixel) ...]
    [(posn? a-pixel) ...]))
```

Função `distance-to-0`

Etapa 2: Como selecionar as coordenadas, caso a entrada seja uma estrutura `posn`? Usando as funções auxiliares `posn-x` e `posn-y`:

```
(define (distance-to-0 a-pixel)
  (cond
    [(number? a-pixel) ...]
    [(posn? a-pixel) ... (posn-x a-pixel) ... (posn-y a-pixel) ...]))
```

Função `distance-to-0`

Etapa 3: Definir como calcular a função em cada caso:

```
(define (distance-to-0 a-pixel)
  (cond
    [(number? a-pixel) a-pixel]
    [(posn? a-pixel) (sqrt
      (+ (sqr (posn-x a-pixel))
        (sqr (posn-y a-pixel))))])])
```

Desenvolvendo Funções com Dados Mistos

- Análise e projeto de dados: determinar quantas e quais classes distintas de dados fazem parte do problema. Como resultado, a definição de dados pode ter várias cláusulas enumerando os tipos de dados do problema;
- Construção do *template*: como a função a ser definida pode operar sobre vários tipos de dados, o corpo da função é uma expressão `cond`, com tantas cláusulas quantos são os tipos de dados envolvidos no problema;
- Definição da função: usando o condicional, o problema é dividido em vários sub-problemas. Assim, podemos tratar cada cláusula do `cond` separadamente resolvendo o problema para o tipo de dados tratado nesta cláusula.

As outras fases (contrato, objetivo, cabeçalho, exemplos, testes) não sofrem modificações pelo uso de dados 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:

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 aos quais a função pode ser aplicada
Template	formular um esboço para a função	colocar uma estrutura cond com uma linha para cada tipo de dados envolvido na definição dos dados do problema. Para identificar cada um dos tipos de dados, usar os predicados de identificação de tipos

Fase	Objetivo	Atividade
Corpo	completar a definição da função	construir expressões Scheme para cada uma das cláusulas do <code>cond</code>
Testes	encontrar erros	aplicar a função aos exemplos e checar se os resultados são os esperados.

Exemplo: Função que calcula o perímetro de uma forma

```
;; Data Definition:  
(define-struct circle (center radius))  
(define-struct square (nw length))  
;; A shape is either  
;; 1. a structure: (make-circle p s)  
;;    where p is a posn, s a number;  
;; 2. a structure: (make-square p s)  
;;    where p is a posn, s a number.  
  
;; Contract, Purpose, Header:  
;; perimeter : shape -> number  
;; to compute the perimeter of a-shape  
  
;; Examples: see tests
```

```
;; Template:
;; (define (f a-shape)
;;   (cond
;;     [(square? a-shape)
;;      ... (square-nw a-shape) ... (square-length a-shape) ...]
;;     [(circle? a-shape)
;;      ... (circle-center a-shape) ... (circle-radius a-shape) ...]))

;; Definition:
(define (perimeter a-shape)
  (cond
    [(circle? a-shape)
     (* (* 2 (circle-radius a-shape)) pi)]
    [(square? a-shape)
     (* (square-length a-shape) 4)]))
```

```
;; Tests: (same as examples)
(= (perimeter (make-square ... 3)) 12)
(= (perimeter (make-circle ... 1)) (* 2 pi))
```

Exercício

Desenvolva uma estrutura e definições de dados para uma coleção de animais de um zoológico. A coleção inclui

aranhas : cujos atributos relevantes são o número atual de pernas (assumimos que elas podem perder pernas ao longo da vida devido a acidentes) e o espaço necessário no caso de transporte;

elefantes : cujo único atributo é o espaço necessário para transporte;

macacos : cujos atributos são o quociente de inteligência e o espaço necessário para transporte.

Construa a função `fits?`, que, dado um animal do zoológico e o volume de uma jaula, determina se a jaula é grande suficiente para transportar o animal em questão.