Estruturas

Fundamentos de Algoritmos

INF05008

Dados Compostos

- A **entrada** (elemento do domínio) para uma função é, em geral, um dado que representa um **objeto** e seus **atributos**
- Exemplo: entrada pode ser informação sobre um CD (artista, gravadora, nome, preço)
- Vários dados são **compostos** em um único dado
- Scheme oferece várias formas de compor dados, entre elas: **estruturas**

Estruturas

- Coordenadas da tela
  - A tela do computador é formada por *pixels*
  - Cada *pixel* possui uma coordenada x e outra y
  - Os pacotes de ensino do DrScheme representam um *pixel* com uma estrutura *posn*
  - Uma estrutura *posn* é criada com a operação *make-posn*, a qual consome dois números (coordenadas x e y)

```
(make-posn 3 4)    (make-posn 8 6)    (make-posn 5 12)
```

Estruturas: Exemplo

Considere uma função que computa a distância de um pixel para a origem:

```
;; distância-para-0 : posn  ->  número
;; Calcula a distância do ponto um-posn até a origem

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

A função `distância-para-0` **consome um único valor** - uma estrutura *posn* - e **produz um único valor** - um número.

Exemplos de uso de `distância-para-0`

```
    (distância-para-0 (make-posn 0 5))
= 5
```

e

```
    (distância-para-0 (make-posn 7 0))
= 7
```

A distância de uma posição com coordenadas x e y para a origem é dada por $\sqrt{x^2 + y^2}$. Logo,

```
   (distância-para-0 (make-posn 3 4))
= 5

   (distância-para-0 (make-posn 8 6))
= 10

   (distância-para-0 (make-posn 5 12))
= 13
```

- Para **acessar** cada coordenada de um pixel, necessitamos de **funções**
- Scheme fornece duas funções para acessar coordenadas de *posn*: `posn-x` e `posn-y`
- Exemplos que mostram a relação dessas operações com `make-posn`:

```
    (posn-x (make-posn 7 0))
= 7
```

e

```
    (posn-y (make-posn 7 0))
= 0
```

Podemos nos referir a uma estrutura *posn* por um **nome** :

```
(define ponto (make-posn 7 0))
```

E temos então:

```
  (posn-x ponto)
= 7
  (posn-y ponto)
= 0
  (* (posn-x ponto) 7)
= 49
  (+ (posn-y ponto) 13)
= 13
```

Podemos completar a definição de *distância-para-0*:

```
(define (distância-para-0 um-posn)
 ... (posn-x um-posn) ...
 ... (posn-y um-posn) ...)
```

E, finalmente, chegamos a:

```
(define (distância-para-0 um-posn)
  (sqrt
    (+ (sqr (posn-x um-posn))
       (sqr (posn-y um-posn)))))
```

Definição de Estrutura

- Usamos estruturas *posn*
- *posn* combina 2 números e é **útil para representar pixels**
- Scheme permite a programadores **definir outras estruturas**

- Definição da estrutura *posn*:

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

- Após a avaliação dessa definição, 3 operações são **automaticamente criadas**:
  - Um **construtor** `make-posn`, que cria uma estrutura `posn`
  - Um **seletor** `posn-x`, que seleciona o valor da coordenada x de uma estrutura `posn`
  - Um **seletor** `posn-y`, que seleciona o valor da coordenada y de uma estrutura `posn`

- Exemplo: `(define-struct entrada (nome cep fone))`
- Estrutura representa uma **entrada** no guia telefônico
- Cada entrada combina os seguintes **valores**/**campos**/**atributos**:
  - Nome,
  - CEP, e
  - Telefone

- 4 operações disponíveis:
  - `make-entrada` (Construtor)
  - `entrada-nome` (Seletor)
  - `entrada-cep` (Seletor)
  - `entrada-fone` (Seletor)

Exemplo de uso:

```
(make-entrada 'PedroRosa 15270 '606-7771)
```

Cria uma estrutura *entrada* com `'PedroRosa` no campo *nome*, `15270` no campo *cep* e `'606-7771` no campo *fone*

nome	*cep*	*fone*
'PedroRosa	15270	'606-7771

Exemplos:

```
    (entrada-nome (make-entrada 'PedroRosa 15270 '606-7771))
= 'PedroRosa
```

Se dermos um nome para uma entrada,

```
    (define guia (make-entrada 'PedroRosa 15270 '606-7771))
```

podemos usar os seletores da seguinte forma:

```
    (entrada-nome guia)
= 'PedroRosa

    (entrada-cep guia)
= 15270
```

Considere a seguinte estrutura:

```
(define-struct estrela (sobrenome nome instrumento vendas))
```

Quais são as operações automaticamente definidas para a estrutura *estrela*?

```
(define-struct estrela (sobrenome nome instrumento vendas))
```

Após essa definição, temos 5 operações:

- `make-estrela`,
- `estrela-sobrenome`,
- `estrela-nome`,
- `estrela-instrumento`, e
- `estrela-vendas`

Criando estruturas `estrela`

```
(make-estrela 'Friedman 'Dan 'ukelele 19004)

(make-estrela 'Talcott 'Carolyn 'banjo 80000)

(make-estrela 'Harper 'Robert 'gaita 27860)
```

Exercício 6.3.2. Considere a seguinte definição de estrutura:

```
(define-struct filme (título produtor))
```

e avalie as seguintes expressões:

```
(filme-título (make-filme 'ThePhantomMenace 'Lucas))

(filme-produtor (make-filme 'TheEmpireStrikesBack 'Lucas))
```

- Funções podem **consumir** e **produzir** estruturas
- Definir uma função que some 20000 às vendas de uma `estrela`

```
;; incrementa-vendas : estrela -> estrela
;; Dada uma estrela, produz a mesma estrela
;; com 20000 a mais em vendas

(define (incrementa-vendas uma-estrela) ...)
```

Exemplo de uso da função `incrementa-vendas`:

```
(incrementa-vendas (make-estrela 'Abba 'John 'vocais 12200)
```

deve produzir:

```
(make-estrela 'Abba 'John 'vocais 32200)
```

```
;; incrementa-vendas : estrela -> estrela
;; Dada uma estrela, produz a mesma estrela
;; com 20000 a mais em vendas

(define (incrementa-vendas uma-estrela)
   (make-estrela (estrela-sobrenome uma-estrela)
                 (estrela-nome uma-estrela)
                 (estrela-instrumento uma-estrela)
                 (+ (estrela-vendas uma-estrela) 20000)))
```

Definição de Dados

Considere a seguinte expressão:

```
(make-posn 'Albert 'Meyer)
```

Ela gera uma estrutura *posn* a partir de 2 símbolos

O quê acontece se aplicarmos `distância-para-0` a esta estrutura?

```
   (distance-to-0 (make-posn 'Albert 'Meyer))
= (sqrt
     (+ (sqr (posn-x (make-posn 'Albert 'Meyer)))
        (sqr (posn-y (make-posn 'Albert 'Meyer)))))
= (sqrt
     (+ (sqr 'Albert)
        (sqr (posn-y (make-posn 'Albert 'Meyer)))))
= (sqrt
     (+ (* 'Albert 'Albert)
        (sqr (posn-y (make-posn 'Albert 'Meyer)))))

ERRO NA EXECUÇÃO !!!
```

Uma **definição de dados** estabelece, em uma mistura de Scheme e Português,

1. Como **construir** uma estrutura
2. Como **usá-la apropriadamente**

Exemplo para *posn*:

Um posn é uma estrutura

```
(make-posn x y)
```

onde 'x' e 'y' são números

Exemplo para estrutura *estrela*:

Uma estrela é uma estrutura

```
(make-estrela sobrenome nome instrumento vendas)
```

onde 'sobrenome', 'nome', e 'instrumento' são símbolos e 'vendas' é um número.

- Esperamos que programadores e usuários **respeitem a definição de dados**
- Obviamente, uma definição de dados **não é uma garantia** de que a definição de dados será obedecida
- Caso não seja, ocorre um **erro de execução**

Projeto de Funções com Estruturas

Função que processa registros de estudantes. As informações importantes sobre estudantes são:

- Nome
- Sobrenome
- Nome do professor titular

```
(define-struct aluno (sobrenome nome professor))
```

Definição de dados para *aluno*:

Um aluno é uma estrutura

```
(make-aluno sobrenome nome professor)
```

onde 'sobrenome', 'nome' e 'professor' são símbolos.

Exemplos de estruturas *aluno*:

```
(make-aluno 'Silva 'José 'Belchior)

(make-aluno 'Cardozo 'João 'Lucélia)

(make-aluno 'Lino 'Mateus 'Lucélia)
```

O novo componente do projeto é o *template* para funções que têm como entrada estruturas *aluno*:

```
;; processa-aluno :  aluno ???  ->  ???

(define (processa-aluno um-aluno ...)
... (aluno-sobrenome um-aluno) ...
... (aluno-nome um-aluno) ...
... (aluno-professor um-aluno) ...)
```

Esse *template* pode ser usado para todas as funções que processam *aluno*!

"Desenvolva uma função que, dados um aluno e o nome de um professor, produza o mesmo aluno trocando o nome do seu professor, caso esse nome seja '`Jedeão`, pelo nome do professor dado."

Quais são os dados que devem constar na definição da estrutura?

```
;; Definição e análise de dados:

(define-struct aluno (sobrenome nome professor))

;; Um aluno é uma estrutura:
;;  (make-aluno sobrenome nome professor)
;; onde 'sobrenome', 'nome' e 'professor' são símbolos
```

Qual será o uso dessa estrutura, de acordo com o enunciado?

"Desenvolva uma função que, dados um aluno e o nome de um professor, ***produza o mesmo aluno trocando o nome do seu professor, caso esse nome seja 'Jedeão, pelo nome do professor dado*** *."*

Como ficam o contrato e o objetivo da função que realiza esta operação? Quais exemplos devemos incluir?

```
;; Contrato: subst-professor : aluno símbolo -> aluno

;; Cria um 'aluno' com novo nome de professor, 'um-professor',
;; se o professor atual for 'Jedeão'

;; Exemplos:
;; (subst-professor (make-aluno 'Lívio 'Tito 'Jedeão) 'Elisa)
;; =
;; (make-aluno 'Lívio 'Tito 'Elisa)
;;
;; (subst-professor (make-aluno 'Aquino 'Jorge 'Amanda) 'Elisa)
;; =
;; (make-aluno 'Aquino 'Jorge 'Amanda)
```

```
;; Template:
;; (define (processa-aluno um-aluno ???)
;; ... (aluno-sobrenome um-aluno) ...
;; ... (aluno-nome um-aluno) ...
;; ... (aluno-professor um-aluno) ...)

;; Definição:
(define (subst-professor um-aluno um-professor)
   (cond
     [(symbol=? (aluno-professor um-aluno) 'Jedeão)
       (make-aluno (aluno-sobrenome um-aluno)
                   (aluno-nome um-aluno)
                   um-professor)]
     [else um-aluno]))
```

Como tornar esta função mais genérica?

```
;; subst-professor : aluno símbolo símbolo -> aluno
;; Cria um 'aluno' com novo nome de professor, 'novo-prof',
;; se o nome do professor atual for 'antigo-prof'
(define (subst-professor um-aluno antigo-prof novo-prof)
   (cond
     [(symbol=? (aluno-professor um-aluno) antigo-prof)
       (make-aluno (aluno-sobrenome um-aluno)
                   (aluno-nome um-aluno)
                   novo-prof)]
     [else um-aluno]))
```

Exercício: 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, crie uma estrutura professor conforme a definição de dados abaixo:

Um professor é um estrutura do tipo:

```
(make-professor nome turma nível)
```

onde 'nome' é uma string, 'turma' é um número e 'nível' é um símbolo