CS61A Lecture 13

2011-07-12
Colleen Lewis

get and put

Demo

get and put

STk> (put 'richmond 'berkeley "$1.75)
ok
STk> (get 'richmond 'berkeley)
$1.75
STk> (get 'berkeley 'richmond)
#f

The reverse isn’t automatically added. Returns #f if there is no entry

get and put

We can add lots of things with the same first word

get and put

STk> (put 'berkeley 'fremont "$4.30)
ok
STk> (get 'berkeley 'fremont)
$4.30
STk> (get 'fremont 'berkeley)
#f

get and put

(put 'berkeley 'richmond "$1.75)
(put 'richmond 'berkeley "$1.75)
(put 'berkeley 'fremont "$4.30)
(put 'fremont 'berkeley "$4.30)
(put 'berkeley 'SFO "$8.65)
(put 'SFO 'berkeley "$8.65)

(define (bart-cost stop1 stop2)
  (get stop1 stop2))

get and put

STk> (bart-cost 'fremont 'berkeley)
$4.30
STk> (bart-cost 'fremont 'SFO)
#f
**Put creates a table**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Berkeley</td>
<td>Fremont</td>
<td>$4.30</td>
</tr>
<tr>
<td>Richmond</td>
<td>Berkeley</td>
<td>$1.75</td>
</tr>
<tr>
<td>SFO</td>
<td>Berkeley</td>
<td>$8.65</td>
</tr>
<tr>
<td>Fremont</td>
<td>Berkeley</td>
<td>$4.30</td>
</tr>
<tr>
<td>SFO</td>
<td>$10.55</td>
<td></td>
</tr>
<tr>
<td>Richmond</td>
<td>$4.85</td>
<td></td>
</tr>
</tbody>
</table>

**How do I add more Bart Stops?**

\[
\text{(define (bart-cost stop1 stop2)}
\quad \text{(get stop1 stop2)})
\]

A) Modify the function `bart-cost`.
B) Call `put` for more stops
C) All of the above
D) None of the above
E) ??

**Tagged Data**

**Removing Ambiguity – Tagging data**

- What does this represent?
  A) \(\frac{3}{4}\)
  B) \(3+4i\)
  C) 3x4 rectangle

"The next thing is a fraction where the car is the numerator and the cdr is the denominator"

**attach-tag**

\[
\text{(define (attach-tag type contents)}
\quad \text{(cons type contents))}
\]

**Selectors**

\[
\text{(define (type-tag tagged-data)}
\quad \text{(car tagged-data))}
\]

\[
\text{(define (contents tagged-data)}
\quad \text{(cdr tagged-data))}
\]
(define (make-square side)
  (attach-tag 'square side))

(define square3 (make-square 3))
(define square5 (make-square 5))

(define (make-circle side)
  (attach-tag 'circle side))

(define circle4 (make-circle 3))
(define circle6 (make-circle 5))

Write area to work for circles and squares

(define (area shape)
  How many times did you call type-tag?
  A)1  B)2  C)3  D)4  E)5

A) ((make-square 4) 'area)
  16

B) (area (make-square 4))
  16

C) (area (contents (make-square 4)))
  16

What is returned by make-square?

A) A shape
B) A pair, where the car is the word square.
C) Tagged data
D) A function

Could you re-write area?

(put 'square 'area
  (lambda (s) (* s s)))
(put 'circle 'area
  (lambda (r) (* pi r r)))

STk> (area (make-square 3))
  9

Show the answer with:
A) Live coding    B) PowerPoint    C) Chalk
How do I add another shape?

A) Modify the function area2.
B) Call put for more shapes
C) All of the above
D) None of the above
E) ??

Error checking: operate

(define (operate op obj)
  (let ((proc (get (type-tag obj) op)))
    (if proc
      (proc (contents obj))
      (error "Unknown op for type")))))

(define (area3 shape)
  (operate 'area shape))

area data directed solution w/out and w/ put

(define (area shape)
  (cond
    ((eq? (type-tag shape) 'square)
     (* (contents shape) (contents shape)))
    ((eq? (type-tag shape) 'circle)
     (* pi (contents shape) (contents shape)))
    (else (error "Unknown shape --- AREA"))))

(define (area3 shape)
  (operate 'area shape))

Message passing make-square

(define (make-square-mp side)
  (lambda (message)
    (cond
      ((eq? message 'area)
       (* side side))
      ((eq? message 'perimeter)
       (* 4 side))
      (else (error "unknown msg"))))
How do you call area?

A) STk> ((make-square-mp 4) 'area)
   16

B) STk> (area (make-square-mp 4))
   16

C) STk> (area (contents (make-square-mp 4)))
   16

Easier way to call area

(define (make-square-mp side)
  (lambda (message)
    (cond
      ((eq? message 'area)
        (* side side))
      ((eq? message 'perimeter)
        (* 4 side))
      (else (error "unknown msg"))))
)(define (area shape)
  (shape 'area))

Message passing versus data directed?

The constructors for this style return tagged data:
A) Data Directed
B) Message Passing
C) Both
D) Neither

Message passing versus data directed?

This method CAN use put/get tables
A) Data Directed
B) Message Passing
C) Both
D) Neither

Message passing versus data directed?

This method used put/get tables in today’s lecture
A) Data Directed
B) Message Passing
C) Both
D) Neither

Message passing versus data directed?

The constructors for this style return a function:
A) Data Directed
B) Message Passing
C) Both
D) Neither
Message passing versus data directed?

If you don’t use `put`, there are a bunch of `cond` cases to handle each type of data in any “operation” e.g. `area`:

A) Data Directed  
B) Message Passing  
C) Both  
D) Neither

Solutions

`area solution`

```scheme
(define (area shape)
  (cond
    ((eq? (type-tag shape) 'square) (* (contents shape) (contents shape)))
    ((eq? (type-tag shape) 'circle) (* pi (contents shape) (contents shape)))
    (else (error "Unknown shape --- AREA"))))
```

`area2 solution`

```scheme
(put 'square 'area
  (lambda (s) (* s s)))
(put 'circle 'area
  (lambda (r) (* 3.1415 r r)))

(define (area2 shape)
  ((get (type-tag shape) 'area) (contents shape)))
```

SOLUTION count-pairs

```scheme
STk>(count-pairs (list 1 2 3 4))
4
STk>(count-pairs (list (cons 1 2) 3 4))
4
(define (count-pairs struct)
  (if (not (pair? struct))
      0
      (+ 1 (count-pairs (car struct))
           (count-pairs (cdr struct)))))
```