

CSC151.02 2014S, Class 08: Documenting Programs and Procedures

Overview

- Preliminaries.
 - Admin.
 - Upcoming work.
 - Questions.
 - Quiz.
- The need for documentation.
- The Six P's - a strategy for documenting procedures.
- A few additional P's.
- Practice.

Preliminaries

Admin

- Today's quiz: Numbers, Drawings, and Procedures.
- Extra credit:
 - Convo Feb. 5., Wednesday, Noon, JRC 101, Free Lunch
 - Why go to convo?
 - Tonight's [fill in your own description] celebration, 5:30-8:00, Harris
 - Home Tennis matches in Field House Saturday, 8:30 a.m. and 3:30 p.m.
 - Others?

Upcoming Work

- Monday's reading: Testing Procedures.
- Documentation writeup (due Monday): Document the `rectangle` procedure from exercise 5 of the procedures lab.
 - Email subject: CSC 151 Writeup 5: Documentation (YOUR NAME(s) HERE)

Questions

What does a procedure definition look like?

```
(define PROC
  (lambda (PARAMS)
    EXPRESSION))
```

For example.

```
(define munge
  (lambda (x)
    (+ x (* x x))))

(define average
  (lambda (x y)
    (/ (+ x y) 2)))
```

How can we think about a procedure call?

Approach 1: Replacement. In the body of the procedure, replace every copy of the parameter with the corresponding value in the call.

```
(munge 5) => (+ 5 (* 5 5))
(munge 7) => (+ 7 (* 7 7))
(munge (- 2 5)) => (munge -3) => (+ -3 (* -3 -3))
(average 3 4) => (/ (+ 3 4) 2)
```

Approach 2: Implicit defines. Implicitly add a `define` statement for each parameter using the corresponding value in the call, then evaluate the body, then forget the definition.

```
(munge 5) => (define x 5)
              (+ x (* x x))
              [forget the definition]
(munge 7) => (define x 7)
              (+ x (* x x))
              [forget the definition]
(average 3 4) => (define x 3)
                  (define y 4)
                  (/ (+ x y) 2)
                  [forget the definitions]
```

Reality is closer to the second than the first, but it doesn't really matter.

Can you explain the relationship between shifting and scaling?

Behind the scenes, we represent the basic drawings with a few values type, color, left, top, width, height

When we hshift a drawing, we add to the left

When we vshift a drawing, we add to the top

When we hscale a drawing, we multiply the width AND we multiply the left

When we vscale a drawing, we multiply the height AND we multiply the top

When we scale a drawing, we multiply width AND height AND left AND top

What happens to the center? Is it also scaled?

The x coordinate of the center is $(+ \text{left} (/ \text{width } 2))$

The x coordinate of the scaled center is $(+ (* \text{scale left}) (/ (* \text{scale width } 2)))$

The x coordinate of the scaled center is $(+ (* \text{scale left}) (* \text{scale} (/ \text{width } 2)))$

The x coordinate of the scaled center is $(* \text{scale} (+ \text{left} (/ \text{width } 2)))$

Conclusion: Yes.

What if I want to scale the object and keep the top and the left the same?

Option 1: Scale it, figure out how much it moved, move it back

Option 2: Write our own procedure that takes advantage of the underlying representation and just scale width and height.

Option 3: Move it so that the top is 0 and the left is 0. Then scale, then move it back

Quiz!

STAY IN YOUR SEATS. YOUR MENTORS WILL HELP COLLECT THE EXAMS!

If you finish early, think about how you would document a `neighbor` procedure that creates a copy of a drawing immediately to the right of the original drawing.

Practice

- Goal: Document *what* our procedure is supposed to accomplish. (*How* is secondary.)
- Approach:
 - Person 1: Suggest one of the P's
 - Person 2: Explain what it does (or say that they may be mistaken)
 - Person 3: Give text to accompany the P
- The problem: Document a procedure that takes a drawing as an input and creates a copy of the drawing that falls immediately to the right.

```
;;; Procedure: [Say what the name is]
;;; neighbor
;;; Parameters: [The inputs to the procedure.  name and type]
;;; d, a drawing
;;; Purpose: [Describe what the procedure does]
;;; Create a drawing directly to the right.
```

```
;;; Produces: [The output. Name and type]a
;;; shape, a drawing
;;; Preconditions: [Requirements for the procedure to work]
;;; [Implicit: We have to be able to use the standard drawing procedures.]
;;; ["It has to be a drawing."]
;;; [Questionable: If we want to ensure that shape is renderable on
;;; an image, we might require something about the bounds of d.]
;;; [No additional.]
;;; Postconditions: [Requirements for the procedure's output.]
;;; (drawing-width shape) = (drawing-width d)
;;; (drawing-height shape) = (drawing-height d)
;;; shape has the same appearance as d
;;; (drawing-left shape) = (drawing-right d)
;;; (drawing-top shape) = (drawing-top d)
```

Samuel A. Rebelsky, rebelsky@grinnell.edu

Copyright (c) 2007-2014 Janet Davis, Samuel A. Rebelsky, and Jerod Weinman. (Selected materials are copyright by John David Stone or Henry Walker and are used with permission.)



This work is licensed under a Creative Commons Attribution 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/> or send a letter to Creative Commons, 543 Howard Street, 5th Floor, San Francisco, California, 94105, USA.