Auxiliar Ocho - CC41A

Lenguajes de Programación

My new religion: Pure functional programming!!!

Oscar E. A. Callaú
oalvarez@dcc.uchile.cl

Santiago - Chile, 13/Oct/2008

1. Benancio de nuevo!!!

Nuestro querido amigo Benancio, esta teniendo problemas con su implementación de *lazyness* y le comenta que tiene la siguiente definición:

```
(define-type CFAE/L-Value
  [numV (n number?)]
  [closureV (param symbol?)
            (body CFAE/L?)
            (env Env?)]
  [exprV (expr CFAE/L?)])
```

1. Dada las expresiones:

   ```
   {with {a 5} {{fun {x} {+ x x}} {+ a 5}}}
   ```

   ```
   {with {a 5} {{fun {x} {+ x x}} {/ a 0}}}
   ```

   Cuales son los resultados que daria el interprete de Benancio?

2. Que errores puede encontrar en la implementación de Benancio?

2. Transparencia referencial

- Definealo en sus palabras.
- Que lenguaje conoce que cumple su definición?
- De ejemplos de lenguajes y expresiones del mismo donde no se cumple este principio.

3. Poder al Estado!!!

En las clases ud. a podido apreciar el uso de las funciones `box`;

```
box      :: A -> Box
box?     :: Box -> Bool
un-box   :: Box -> A
set-box! :: Box x A -> A
```

que nos permiten tener variables mutables (su estado pueda variar en el tiempo). Usando `box` implemente:

counter , Implementa un conjunto de funciones que permita manejar una estructura contador, ej.:

```
> (define c (new-counter))
> (inc c)
1
> (inc c)
2
> (dec c)
1
...
```

Stack , Usando la misma técnica anterior implemente una pila con las funciones::

(new-stack) , retorna una pila vacía.

(push stack val) , agrega `val` a la pila `stack`.

(pop stack) , saca el elemento de encima de la pila y lo retorna.

(peek stack) , retorna el valor encima de la pila `stack`, sin modificarla.