Taller 2: Semántica en los lenguajes de programación. Fundamentos de Lenguajes Programación

Carlos Andres Delgado S, Ing *

Abril de 2017

1. Interpretador a construir

Descargue el interpretador del taller y construya las expresiones. En el interpretador se le dejan algunos ejemplos para que pruebe, los cuales dan los siguientes resultados, los cuales son indicados por $->$. En este interpretador debe pensar una forma de implementar los ambientes de tal forma:

- Inicialmente el ambiente está vacío
- A medida que vamos ejecutando sentencias de asignación o creación de variables el ambiente cambia (los programas no son independientes como es el caso del interpretador de clase)
- Debe crearse un ambiente extendido en las sentencias **local**

Los pasos de parámetros el taller son por referencia.

```
x
-> Error x no existe
X
-> Error X no existe
4
-> 4
true
-> true
4.5
-> 4.5
'perro'
-> 'perro'
var x = 8
-> #void
x
-> 8
set x = 3
-> #void
set y = 4
-> Error y no existe
x
-> 3
{ 1 2 3 4 5}
-> { 1 2 3 4 5}
#{ 1 2 3 4 5}
-> #{ 1 2 3 4 5}
+(4,2)
-> 6
+(4,2,2,3,4,5)
-> 20
>=(4,2)
-> true
if >=(4,2) ? 3 : 8 :
-> 3
fun(factorial x) if ==(x,0) ? 1 : (factorial -(x,1)) end
```

*carlos.andres.delgado@correounivalle.edu.co

```
-> #void
(factorial 5)
-> 120
fun(funcion x y) set x = +(x,y) set y = +(y,3) *(x,y) end
-> #void
(fact 4)
-> Error la función fact no existe
var fact = proc(x) if ==(x,0) ? 1 : (fact -(x,1)) end
-> #void
(fact 4)
-> 24
var func = proc(x y) set x = +(x,y) set y = +(y,3) *(x,y) end
-> #void
size({1 2 3 4 5})
-> 5
size(#{2 3 4 5})
-> 4
nth({1 2 3 4 5}, 3)
-> 3
nth(#{2 3 4 5}, 2)
-> 3
agregar(#{1 2 3 4 5}, 10)
-> #{1 2 3 4 5 10}
delete(#{1 2 3 4 5}, 2)
-> #{1 3 4 5}
x
-> 3
var y = 5
-> #void
cond [>(x,y) 1] [<(x,y) 2] else 2
-> 1
if <(x,y) ? 'hola' : 'no'
-> 'no'
local ( ( a = 3 b = 3) (+(a,b,3)) )
-> 9
local ( ( a = 3 b = 3) ('prueba') )
-> 'prueba'
//Los pasos del taller son por referencia
fun(aplicar x) set x = 10 'exito' end
-> #void
var a  = 3
-> #void
a
-> 3
(fun a)
-> 'exito'
a
-> 10
```

Es importante **que haga el control de errores con eopl:error**

Bugs

La sentencia **cond** no puede trabajar con textos, se queda un bucle infinito.

Aclaraciones

1. El taller es en grupos de máximo tres (3) estudiantes.
2. La solución del taller debe ser subida al campus virtual en la fecha especificada por el docente.
3. Las primeras líneas de cada archivo deben contener los nombres y códigos de los estudiantes.

```
;; Pepito perez 123123
;; Juanito no hiceNada 231321
;; Le Pague AOtro 2131232
```

4. Para la construcción de funciones utilice expresividad de **lambda** y use la función **eopl:error** para el reporte de errores, no deje que el Dr Racket lo haga por usted.
5. Comente el código explicando que hizo en cada parte, ejemplo:

```
(define eval-expression
       (lambda exp ambiente)
          (cases exp
            ;;si es una variable busca en el ambiente su valor
            (variable-exp (variable) (buscar-variable ambiente)
            ;;Si es un número retorna su valor
            (entero-exp (dato) dato)
            ...
```

La ausencia de comentarios rebajará su nota.