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

Carlos Andres Delgado S, Ing [*]

Abril de 2018

1. Reglas del taller

1. El taller debe ser entregado el **Sábado, 14 de Abril de 2018 08:00am** hora de Colombia, por el campus virtual. El enlace se cierra a esa hora y no le permitirá enviar después de eso.
2. El código debe ir comentado explicando brevemente cada función y también debe incluir lo solicitado en los puntos. Si no incluye esto, se sanciona 0.5 nota del taller.
3. No se permite división de grupos ni cambios de integrantes de última hora. No se recibirá el taller si esto va producir conflictos entre los integrantes.
4. Debe entregar el código fuente organizado en carpetas dentro del primer nivel del archivo comprimido, no cree una jerarquía compleja difícil de revisar. Si no realiza este paso se sanciona 0.2 en la nota. Lo importante es que eso sea claro para la revisión.
5. No se permite copiar código de Internet ni de sus compañeros. Si se encuentra código copiado el taller será anulado por completo.
6. Entregue un sólo archivo comprimido. No entregue archivos comprimidos dentro de archivos comprimidos, ya que esto dificulta la revisión enormemente. Si hace esta mala práctica se aplica una sanción con 0.5 en la nota del taller.
7. No deje que el enlace del campus se cierre. No se reciben trabajos por correo, no insista. Si el campus falla, esto será verificado con la DINTEV. Recuerde estar autenticado ya que el curso permite acceso a invitados
8. Las primeras líneas de cada archivo de código fuente, deben tener los integrantes del grupo con sus nombres y código completos. Si no cumple esto, será sancionado con 0.5 en la nota del taller.

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

2. Interpretador a construir

Importante: Para este punto entregue un sólo archivo, denominado **interpretador.rkt**. No se entrega informe, sólo el código.

Tenga en cuenta que se deben utilizar ambientes en este interpretador.

- Inicialmente el ambiente está vacío
- Debe crearse un ambiente extendido en las sentencias **local** y para evaluar una función

Importante: El $->$ indica la respuesta que da el comando.

```
x
-> Error x no existe
4
-> 4
-8
-> -8
verdadero
-> verdadero
falso
-> falso
4.5
-> 4.5
-4.5
-> -4.5
'perro'
-> 'perro'
[1, 2, 3, 4, 5]
-> list(1 2 3 4 5)
+(4,2)
-> 6
+(4,2,2,3,4,5)
-> 20
>=(4,2)
-> 'verdadero
<(4.5,2)
-> 'falso
<=(4,2.3)
-> falso
si >=(4,2) ? 3 : 8 fin
-> 3
fun(x y) si >=(x,4) ? 4 : 8 fin fin
-> Funcion creada
largo([1, 2, 3, 4 5])
-> 5
obtenerElemento([1, 2, 3, 4, 5], 3)
-> 3
local a = 3 b = 3 haga +(a,b,3) fin
-> 9
local a = 3 b = 3 haga 'prueba' fin
-> 'prueba'
local funcion = fun(x y) si >=(x,4) ? 4 : 8 fin fin haga (funcion -1) fin
-> 8
```

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

1. (10 puntos) Realice como comentario en el código la especificación de la gramática en forma BNF
2. (5 puntos) Construya la especificación léxica
3. (5 puntos) Construya la especificación gramatical
4. (10 puntos) Funcionamiento primitivas: Suma +, −, >=, <=,> y <
5. (15 puntos) Funcionamiento las listas: creación y consulta de tamaño y elemento.
6. (10 puntos) Funcionamiento de los condicionales
7. (20 puntos) Funcionamiento de los locales.
8. (25 puntos) Funcionamiento de las funciones.

Explique con comentarios cómo realizó todos los puntos, esto es parte de la nota del punto. La calidad de su explicación determinará la puntuación.

Importante: Diseñe algunas pruebas (minimo 3). Debe tener en cuenta que su nota podría verse afectada si se comenten errores conceptuales o de implementación.