# Lenguajes de programación 2016-1
# Tarea 1

José Ricardo Rodríguez Abreu
Martínez Martínez Julio César
Zamudio Cervantes Luis Enrique
Profesora: Karla Ramírez Pulido
Ayudante: Héctor Enrique Gómez Morales

September 21, 2015
Facultad de Ciencias UNAM

## 1. Problema I

Hemos visto en clase que la definición de sustitución resulta en una operación ineficiente: en el peor caso es de orden cuadrático en relación al tamaño del programa (considerando el tamaño del programa como el número de nodos en el árbol de sintaxis abstracta). También se vio la alternativa de diferir la sustitución por medio ambientes. Sin embargo, implementar un ambiente usando un stack no parece ser mucho mas eficiente.
Responde las siguientes preguntas.

1. Provee un esquema para un programa que ilustre la no-linealidad de la implementación de ambientes basada en un stack. Explica brevemente porque su ejecución en tiempo no es lineal con respecto al tamaño de su entrada.

   Sea P el programa que va de la siguiente manera:
   .  {with $\{x\ 0\}$
   .  {with $\{x_1\ x\}$
   .  {with $\{x_2\ x\}$
   .  ...
   .  {with $\{x_n\ x\}$
   .  $\{+\ x_n\ \{+\ \{x_{n-1}\ \{+\ ...\ \{+\ x_1\ x\}\}\}\}\}...\}\}\}$

   El programa P tiene como variable en cada uno de los nodos del árbol de sintaxix abstracta a la variable $x$. Por cada asignacióin with, el ambiente guarda el valor en el stack. Si cada vez que agregamos una variable nueva a nuestro stack, nos pide (en particular en el programa P, la primer variable declarada en el stack) una antes ya agregada, el peor de los casos $n^2$ sobre el número de variables en nuestro ambiente ya que revisará casa uno de los nodos $n$ veces en la estructura de datos.

2. Describe una estructura de datos para un ambiente que un interprete de FWAE pueda usar para mejorar su complejidad.

   La solución al problema de los ambientes o closure es una estrutura de datos que pueda almacenar apuntadores a subrutinas y funciones más el conjunto de variables disponibles en el momento que llamamos a nuestra variable actual (su ambiente inmediato anterior). De esa manera sabríamos siempre que variable tenemos que usar y cuales podemos disponer. Suponiendo que podamos usar cualquier estructura de datos, propondía usar una tabla hash.
   Siendo el código hash de cada variable su propio id y su valor, el valor almacenado en dicha tabla, sería de tiempo constante (amortizado) encontrar en el ambiente el valor de una variable. Si el valor del ambiente cambia es a lo más lineal el tiempo de creación del nuevo ambiente. Por cada función que se mande a llamar, podemos llevar el último de los ambientes junto con la función y la toma de cualquier valor dentro del ambiente será siempre de tiempo constante.

3. Muestra como usaría el interprete esta nueva estructura de datos.

   Supongamos que tenemos el mismo programa P que usamos en el punto 1. Y supongamos que nuestro Hashcode es n si $x_n$ es la variable que estamos agregando al ambiente.
   Primero tenemos a $x$ con su valor de 0, entonces agregamos a $x$ junto con su valor y una copia de su ambiente actual '().
   En el segundo momento tenemos a $x_1$ y (aplicando evaluación glotona) optenemos el valor de $x$ en tiempo constante (para todas las $x_n$ pasa muy parecido), agregamos a $x_1$ y su valor a nuestro ambiente principal. Al final, en la suma de cada uno de los $x_n$ que hay, optener su valor es de tiempo constante haciendo la evaluación de los más rápido que podemos implementar.

4. Indica cual es la nueva complejidad del interprete (análisis del peor caso) y de forma informal pero rigurosa pruébalo.
   Suponiendo que tenemos a una función que necesite de todas las varibales que llevamos hasta ahora; la complejidad de optener el valor de esa llamada será lineal sobre el número de elementos que tenga nuestra tablahash. Ya que optener el valor de una variable es en tiempo constante, y en el peor de los casos tenemos TODAS las variables que son llamadas por esa función. Así que es lineal.

## 2. Problema II

Dada la siguiente expresión de FWAE:

```
{with {x 4}
      {with {f {fun {y} {+ x y}}}
            {with {x 5}
                  {f 10}}}}
```

debe evaluar a (num 14) usando alcance estático, mientras que usando alance dinámico se obtendroia (num

15), Ahora Ben un agudo pero excéntrico estudiante dice que podemos seguir usando alcance dinámico mientras tomemos el valor mas viejo de x en el ambiente en vez del nuevo y para este ejemplo el tiene razón.

1. ¿ Lo que dice Ben esta bien en general? si es el caso justifícalo.

2. Si Ben esta equivocado entonces da un programa de contraejemplo y explica por que la estrategia de evaluación de Ben podría producir una respuesta incorrecta.

Lo que dice Ben no es cierto por la siguiente razón:
Usemos la siguiente expresión como contraejemplo:

```
.      {with {x 1}
.            {with {x 2}
.                  {with {f {fun {y} {+ x y}}}
.                        {with {x 3}
.                              {with {x 4}
.                                    {f 10}}}}}
```

Nuestro ambiente en forma de lista quedaría algo así: $'(f\ 10, x\ 4, x\ 3, f\ y, x\ 2, x\ 1)$.
Si usaramos la idea de Ben entonces, en teoría, la evaluación siempre sería exactamente igual, pero no es así. Si evaluamos con alcance estático normal, el resultado sería la evaluación de $f\ 10$ con $x = 2$ (ya que empezamos a buscar a $x$ a partir de la definición de la función,) y el resultado sería $f = 12$. Si evaluamos con el método de Ben, el valor de $x$ cambia a $x = 1$ (ya que es el valor más viejo del ambiente) y la evaluación de $f$ cambia a $f = 11$.
Ahora si hacemos la evaluación con alcance dinámico, con el método normal, tomamos el primer valor de $x$ que encontremos arriba del ambiente, siendo la evaluación de $f$ con $x = 4$ sea $f = 14$. Si usamos el método de Ben vemos que la evaluación sigue siendo $f = 11$ por como está nuestro ambiente y como buscamos a nuestro valor de $x$.
Como dimos un contraejemplo para cada forma de evaluación (dinámica y estática) con la idea de Ben, podemos decir que su idea no es correcta.

**3.Problema III**

Dada la siguiente expresión de FWAE con with multi-parametrico:

```
.      {with {{x 5} {adder {fun {x} {fun {y} {+ x y}}}} {z 3}}
.            {with {{y 10} {add5 {adder x}}}
.                  {add5 {with {{x {+ 10 z}} {y {add5 0}}}
.                        {+ {+ y x} z}}}}}
```

1. Da la forma Bruijn de la expresión anterior.

```
.      {with {{5} {{fun {x} {fun {y} {+ <: 0, 0 > y}}}} {3}}
.            {with {{10} {<: 0, 1 > <: 0, 0 >}}
```

. $\{<: 0, 1 > \{\text{with } \{\{+ \ 10 <: 1, 2 >\}\} \ \{<: 0, 1 > 0\}\}$
. $\{+ \ \{+ <: 0, 1 > <: 0, 0 >\} <: 2, 2 >\}\}\}\}$

2. Realiza la corrida de esta expresión, es decir escribe explícitamente cada una de las llamadas tanto para subst y interp, escribiendo además los resultados parciales en sintaxis concreta.