# Lenguajes de Programación

Dr. Carlos Artemio Coello Coello
Tarea No. 4
23 de febrero de 2016

## Pascal

Lea el reporte "Why Pascal is Not My Favorite Programming Language", escrito por Brian W. Kernighan disponible en:

`http://delta.cs.cinvestav.mx/~ccoello/lenguajes/bibleng.html`

En este reporte, Kernighan critica severamente algunas de las decisiones de diseño adoptadas en Pascal, cuando las compara con las de C. Este reporte y lo visto en clase acerca de Pascal le proporcionará el material necesario para responder a las siguientes preguntas:

1. (**10 puntos**) Explique cuál es el "problema más grande con Pascal" (*biggest single problem*), de acuerdo a Kernighan y discuta brevemente cómo podría resolverse.

2. (**10 puntos**) ¿Cómo resuelve Pascal la compartición de información entre variables (recuerde el problema del bloque COMMON en FORTRAN)? ¿Cómo lo resuelve C?

3. (**10 puntos**) Explique cuál es "una de las limitaciones más frustrantes" de Pascal de acuerdo a Kernighan y aún los más acérrimos fanáticos de Pascal? ¿Cómo resuelve C este problema?

4. (**15 puntos**) Aunque Kernighan está de acuerdo en que los *conjuntos* en Pascal son muy eficientes y constituyen una buena idea, hace ciertas críticas a esta estructura de datos. ¿Qué les critica? ¿Está de acuerdo con su punto de vista? ¿Justifica eso el no tener conjuntos en C? Discuta.

5. (**10 puntos**) Discuta algunas de las implicaciones de no tener ciclos con la decisión en medio (o una sentencia como *break* de C, que nos permite salirnos de un ciclo) en Pascal.

6. (**10 puntos**) ¿Coincide con la crítica hecha a la sentencia de casos de Pascal? ¿Por qué es tan malo no tener un caso por omisión (*default*)? Explique.

7. (**20 puntos**) Explique algunas de las razones por las cuales Pascal no es adecuado para programación de sistemas (puede usar algunos de los argumentos de Kernighan, pero debe agregar al menos uno que sea propio). ¿Qué relación guardan los conjuntos con esto?

8. (**15 puntos**) ¿Cuáles son los problemas de diseñar un compilador de una sola pasada en el cual todo tiene que declararse antes de poder usarse? Relacione este asunto con la programación estructurada y explique brevemente el enfoque adoptado por C.

# Programación en Scheme

1. (**20 puntos**) Escriba un procedimiento **hexadecimal->decimal** que tome como entrada una lista que represente un número hexadecimal y que regrese su equivalente en decimal (como valor numérico, no como lista). Su procedimiento debe verificar que el usuario introduzca únicamente letras que sean válidas para un número hexadecimal. Pruebe su procedimiento con los siguientes ejemplos:

   (**hexadecimal->decimal** '(a b c d e f 5)) $\Longrightarrow$ 180150005
   (**hexadecimal->decimal** '(z 5)) $\Longrightarrow$
   Error: el argumento (z 5) no es aceptable
   (**hexadecimal->decimal** '(1 g)) $\Longrightarrow$
   Error: el argumento (1 g) no es aceptable
   (**hexadecimal->decimal** '(1 0)) $\Longrightarrow$ 16
   (**hexadecimal->decimal** '(8 f a 0)) $\Longrightarrow$ 36768
   (**hexadecimal->decimal** '()) $\Longrightarrow$ 0

2. (**20 puntos**) Defina un procedimiento **decimal->hexadecimal**, que será la contraparte del procedimiento anterior. En este caso, el procedimiento tomará como entrada un entero y regresará su equivalente hexadecimal en forma de lista. Pruebe su procedimiento con los siguientes ejemplos:

   (**decimal->hexadecimal** 1215) $\Longrightarrow$ (4 b f)
   (**decimal->hexadecimal** -5) $\Longrightarrow$
   Error: el argumento -5 no es aceptable
   (**decimal->hexadecimal** 4.5) $\Longrightarrow$
   Error: el argumento 4.5 no es aceptable
   (**decimal->hexadecimal** 915) $\Longrightarrow$ (3 9 3)
   (**decimal->hexadecimal** 0) $\Longrightarrow$ (0)

   **NOTA:** Queda estrictamente prohibido usar procedimientos para manipulación de cadenas en la implementación de **hexadecimal->decimal** o de **decimal->hexadecimal**.

3. (**Bonificación: 10 puntos**) En la tarea anterior, implementaron un procedimiento llamado **cuenta-todo-el-fondo** usando procedimientos auxiliares. Re-escriba este procedimiento de tal manera que tenga una definición local que tome a la lista **ls** como su único argumento. Su procedimiento debe tener un solo **define** y debe usar **let** y/o **letrec** para sus definiciones locales. Pruebe su procedimiento con:

   (**cuenta-todo-el-fondo** 'a '((a) b (c a) d)) $\Longrightarrow$ 3
   (**cuenta-todo-el-fondo** 'a '((((b (((a)) c)))))) $\Longrightarrow$ 2
   (**cuenta-todo-el-fondo** 5 '(((((((5) ((6) 5))) 5)))) $\Longrightarrow$ 1
   (**cuenta-todo-el-fondo** 1 '()) $\Longrightarrow$ 0

Le resultará útil leer el capítulo 5 de los apuntes de Scheme que están disponibles en la página web del curso.

La solución a TODOS estos problemas deberá incluirse en un solo archivo ASCII (llamado **sol-tarea4.scm**) en un diskette que se entregará junto con el resto de su tarea. Asegúrese de incluir también una impresión del código fuente de sus procedimientos dentro de su tarea. Las soluciones deben implementarse con los nombres que se indican en cada problema de los antes enumerados.

**Fecha de entrega**: *Jueves 3 de marzo a las 12:00hrs.* Toda tarea entregada tarde será penalizada con 10% (sobre la calificación obtenida) por cada periodo de 24 horas que se retrase su entrega. Es importante hacer notar que las tareas entregadas tarde deberán enviarse por correo electrónico.