Lenguajes de Programación

Dr. Carlos Artemio Coello Coello
Tarea No. 3
2 de febrero de 2016

ALGOL-60

1. Considere el siguiente programa en Algol-60:

```
begin integer i,j; integer array A[1:3];

  procedure P(x,y); integer x, y;
    begin
      y:=2;
      Print(x);
      i:=3;
      Print(x);
      Print(y);
    end

  A[1]:=7; A[2]:=11; A[3]:=13;
  i:=1;
  P(A[i],i);
  P(i,A[i]);
end
```

¿Qué valores imprime este programa para cada una de las situaciones siguientes?

- (**5 puntos**) **x** se pasa por valor; **y** se pasa por valor.
- (**5 puntos**) **x** se pasa por valor; **y** se pasa por nombre.
- (**5 puntos**) **x** se pasa por nombre; **y** se pasa por valor.
- (**5 puntos**) **x** se pasa por nombre; **y** se pasa por nombre.

Notación BNF Extendida

1. (**10 puntos**) Nombre una parte de la sintaxis de la mayor parte de los lenguajes de programación que NO puede ser expresada usando una gramática regular (es decir, que requiere una gramática libre de contexto).

2. (**30 puntos**) Un cierto distribuidor de revistas almacena cada suscripción en una base de datos y les asigna un número de código. Este número de código luce de la siguiente forma:

   ```
   JIP081#03018701890189019001900790078
   ```

   Las tres primeras letras de este código corresponden a las tres primeras letras del apellido del suscriptor. Los siguientes tres caracteres son los tres primeros números de su calle. Este campo se omite si no se proporciona el nombre de la calle (p.ej., si la dirección es un apartado postal). El siguiente caracter es siempre un signo de número (#). Los siguientes dos caracteres indican el número de periodos de la suscripción, el cual puede ser un entero entre 1 y 99. Los siguientes cuatro caracteres indican la fecha de inicio y finalización de la suscripción en el formato **mmaa** (mes, año). Los últimos dos caracteres forman un código de identificación de dos dígitos para la revista.

   En el ejemplo anterior, los campos pueden entonces interpretarse de la manera siguiente:

`JIP`	Primeras tres letras del apellido del suscriptor.
`081`	Primeros tres dígitos de la calle donde se entrega la revista.
`#`	Signo de número.
`03`	Número de periodos de la suscripción.
`0187`	Fecha de inicio del primer periodo.
`0189`	Fecha de finalización del primer periodo.
`0189`	Fecha de inicio del segundo periodo.
`0190`	Fecha de finalización del segundo periodo.
`0190`	Fecha de inicio del tercer periodo.
`0790`	Fecha de finalización del tercer periodo.
`78`	Código de identificación de la revista.

   Proporcione una especificación para un lenguaje que reconozca todas las formas posibles de códigos de suscripción usando BNF pura (no use BNF extendida). Asegúrese de que su lenguaje acepta exactamente lo que se especificó anteriormente—nada más y nada menos. Puede tener que inventar algunos nombres para los símbolos no terminales. Puede también

suponer que `<letra>` y `<dígito>` son símbolos no terminales y que están predefinidos y listos para usarse.

Deberá crear sus propios ejemplos para probar su definición en BNF, de tal forma que cubra todos los casos posibles. Las siguientes dos cadenas son ejemplos del tipo de derivaciones que su descripción en BNF debe permitir:

```
JIP081#030187018905890690059112927 8
SMI#010719128733
```

Lectura

Lea:

Edsger Dijkstra, "Go To Statement Considered Harmful", *Communications of the ACM*, Vol. 11, No. 3, pp. 147–148, March 1968.

- (**5 puntos**) ¿Cómo relacionaría la hipótesis de Sapir-Whorf con las afirmaciones de Dijkstra acerca de la sentencia **goto**? Discuta.
- (**5 puntos?**) Si coincidimos con Dijkstra en que la sentencia **goto** además de dañina es superflua, ¿a qué atribuya que lenguajes modernos como C la incluyan? Discuta.

Lea:

David Cann, "Retire FORTRAN? A Debate Rekindled", *Communications of the ACM*, Vol. 35, No. 8, pp. 81–88, August 1992.

- (**5 puntos**) Considere las 2 vertientes que aborda este artículo. ¿Está a favor o en contra de la afirmación de que lenguajes imperativos como FORTRAN nos limitan en cuanto a nuestro nivel de abstracción (hipótesis de Sapir-Whorf)? Discuta.
- (**5 puntos**) ¿Consideraría que la programación funcional sería una alternativa viable al FORTRAN para aplicaciones de cómputo científico, a pesar de ser esa área precisamente la especialidad de dicho lenguaje? Discuta.

Programación en Scheme

1. (**5 puntos**) Defina un procedimiento **insertar-derecha** que tome como parámetros **nuevo**, **viejo** y **ls**, y que construya una lista obtenida de la

inserción del elemento **nuevo** a la derecha de cada ocurrencia de alto nivel de **viejo** en la lista **ls**. Pruebe su procedimiento con los siguientes ejemplos:

(**insertar-derecha** 'z 'a '(a b a c a)) $\Longrightarrow$ (a z b a z c a z)
(**insertar-derecha** 0 1 '(0 1 0 1)) $\Longrightarrow$ (0 1 0 0 1 0)
(**insertar-derecha** 'dog 'cat '(my dog is fun)) $\Longrightarrow$ (my dog is fun)
(**insertar-derecha** 'dos 'uno '()) $\Longrightarrow$ ()

2. (**5 puntos**) Defina un procedimiento **suma-todo** que encuentre la suma de los números de una lista que puede contener sublistas anidadas de números. Pruebe su procedimiento con los siguientes ejemplos:

(**suma-todo** '((1 3)(5 7)(9 11))) $\Longrightarrow$ 36
(**suma-todo** '(1 (3 (5 (7 (9)))))) $\Longrightarrow$ 25
(**suma-todo** '()) $\Longrightarrow$ 0

3. (**5 puntos**) Defina un procedimiento **cuenta-todos-los-parentesis** que tome como argumento una lista y cuente el número de paréntesis que abren y cierran en dicha lista. Pruebe su procedimiento con los siguientes ejemplos:

(**cuenta-todos-los-parentesis** '()) $\Longrightarrow$ 2
(**cuenta-todos-los-parentesis** '((a b) c)) $\Longrightarrow$ 4
(**cuenta-todos-los-parentesis** '(((a () b) c) () ((d) e))) $\Longrightarrow$ 14

4. (**5 puntos**) Defina un procedimiento **cuenta-todo-el-fondo** que tome como argumentos un elemento **elem** y una lista **ls** y que regrese el número de elementos en **ls** que NO sean iguales que **elem**. Use el predicado más adecuado para comparar igualdad de forma que funcione con los ejemplos indicados a continuación:

(**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** 'b '()) $\Longrightarrow$ 0

5. (**Bonificación: 10 puntos**) Defina un procedimiento ITERATIVO **haz-lista-asc-de-ents** que tome como argumento un entero n y produzca una lista de enteros de 1 a n en orden ascendente. Después escriba otro procedimiento ITERATIVO llamado **haz-lista-desc-de-ents** que tome cualquier entero n como argumento y que produzca una lista de enteros de n a 1 en orden descendente. Pruebe sus procedimientos con:

(**haz-lista-asc-de-ents** 5) $\Longrightarrow$ (1 2 3 4 5)
(**haz-lista-desc-de-ents** 5) $\Longrightarrow$ (5 4 3 2 1)

(**haz-lista-asc-de-ents** 0) $\Longrightarrow$ ()
(**haz-lista-desc-de-ents** 0) $\Longrightarrow$ ()
(**haz-lista-asc-de-ents** 1) $\Longrightarrow$ (1)
(**haz-lista-desc-de-ents** 1) $\Longrightarrow$ (1)

NOTA Para implementar estos procedimientos, puede usar **append**, pero queda estrictamente prohibido usar **reverse**. Adicionalmente, las implementaciones de estos dos procedimientos deben ser independientes entre sí (es decir, no es válido que defina una en función de la otra).

Le resultará útil leer el capítulo 4 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-tarea3.scm**) en un CD 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: *Martes 16 de febrero 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.