Lenguajes de Programación

Dr. Carlos Artemio Coello Coello
Tarea No. 2
26 de enero de 2016

FORTRAN

1. (**5 puntos**) El GOTO de FORTRAN permite hacer ciclos un tanto barrocos. Por ejemplo, es posible efectuar ciclos con la decisión en medio (*mid-decision loops*) como el siguiente:

```
100     ...primera mitad del ciclo...
        IF (condición se cumple) GOTO 200
        ...segunda mitad del ciclo...
        GOTO 100
200     ...
```

   Sugiera un uso práctico para este tipo de ciclos (*mid-decision loops*).

2. (**10 puntos**) Proponga y defienda una solución al problema de GOTO asignado y el GOTO calculado (*assigned GOTO* y *computed GOTO*) de FORTRAN.

3. (**5 puntos**) Considere la siguiente subrutina en FORTRAN:

```
SUBROUTINE TEST(X,Y,Z)
X=1
Z=X+Y
RETURN
END
```

   y considere el siguiente fragmento de código:

```
N=2
CALL TEST(N,N,M)
```

¿Cuál será el valor final de M si se pasan los parámetros por referencia? ¿Cuál será el valor final de M si se pasan los parámetros por valor-resultado?

4. (**10 puntos**) Hay diferentes variantes del mecanismo de paso de parámetros denominado valor-resultado. La dirección del argumento puede calcularse una vez (en la entrada de un subprograma) o dos (en la entrada y en la salida de un subprograma). Describa la salida del siguiente programa bajo cada una de estas dos variantes de este mecanismo de paso de parámetros:

```
DIMENSION A(2)
I=1
A(1)=10
A(2)=11
CALL SUB(I, A(I))
PRINT, A(1),A(2)
END

SUBROUTINE SUB(K,X)
PRINT, X
K=2
X=20
RETURN
END
```

¿Depende la salida del orden en que se copian los resultados?

5. (**15 puntos**) ¿Qué mejoras cree que debiera hacérsele al COMMON de FORTRAN? Explique brevemente los cambios que sugiere y justifíquelos.

6. (**5 puntos**) Sugiera una nueva sintaxis para el ciclo **DO** que sea menos propensa al error que vimos en clase (ver acetato 86 de la clase del 22 de enero de 2016) en el cual el usar un punto en vez de una coma, produce una asignación de variables, cuando el programador quería efectuar un ciclo.

7. Para cada uno de los siguientes principios, liste una (o dos, en algunos casos) característica de FORTRAN que (a) viola ese principio y una que (b) ilustre positivamente ese principio:

   a) (**3 puntos**) Estructural (*Structure Principle*)

   b) (**3 puntos**) Consistencia Sintáctica (*Syntactic Consistency Principle*)

   c) (**2 puntos**) Defensa en Profundidad (*Defense in Depth*) (no tiene que dar un ejemplo positivo en este caso).

d) (**3 puntos**) Preservación de Información (*Preservation of Information*)

e) (**3 puntos**) Abstracción (*Abstraction*)

f) (**3 puntos**) Cero-Uno-Infinito (*Zero-One-Infinity*)

g) (**3 puntos**) Ocultamiento de Información (*Information Hiding*)

8. (**10 puntos**) Derive la ecuación de direccionamiento para arreglos tridimensionales almacenados en modo "columna principal" (*column-major*). Haga luego lo mismo para arreglos tridimensionales almacenados en modo "fila principal" (*row-major*).

9. (**10 puntos**) Tanto los números en doble precisión como los complejos ocupan 2 palabras de memoria en vez de una. Derive la ecuación de direccionamiento para arreglos de una, dos y tres dimensiones de números complejos o de doble precisión.

Programación en Scheme

1. (**5 puntos**) Defina un procedimiento **producto-punto**, que tome como entrada 2 tuplas de la misma longitud, multiplique sus componentes correspondientes, y sume los productos resultantes. Pruebe su procedimiento con los siguientes ejemplos:

   (**producto-punto** '(3 4 -1) '(1 -2 -3)) $\Longrightarrow$ -2
   (**producto-punto** '(0.003 0.035) '(8 2)) $\Longrightarrow$ 0.094
   (**producto-punto** '(5.3e4) '(2.0e-3)) $\Longrightarrow$ 106.0
   (**producto-punto** '() '()) $\Longrightarrow$ 0

2. (**5 puntos**) Defina un procedimiento **indice** que tome dos argumentos: un elemento **a** y una lista **ls**. Este procedimiento debe regresar el índice de **a** en **ls**; es decir, la posición (contando a partir de cero) que ocupa **a** en **ls**. Si el elemento no está en la lista, el procedimiento debe regresar -1. Pruebe su procedimiento con los siguientes ejemplos:

   (**indice** 3 '(1 2 3 4 5 6)) $\Longrightarrow$ 2
   (**indice** 'so '(do re mi fa so la si do)) $\Longrightarrow$ 4
   (**indice** 'a '(b c d e)) $\Longrightarrow$ -1
   (**indice** 'gato '()) $\Longrightarrow$ -1

3. (**5 puntos**) Defina un procedimiento **frente-lista** que tome como argumento una lista de elementos **ls** y un entero positivo **num** y regrese los primeros **num** elementos de alto nivel de **ls**. Si **num** es mayor que el número de elementos de alto nivel de **ls**, deberá producirse un mensaje de error (puede usar el procedimiento **error** definido en el capítulo 3 de las notas de Scheme asignadas por el instructor). Pruebe su procedimiento con los siguientes ejemplos:

(**frente-lista** '(a b c d e f g) 4) $\Longrightarrow$ (a b c d)
(**frente-lista** '(a b c) 4) $\Longrightarrow$ Error: la longitud de (a b c) es menor que 4
(**frente-lista** '(a b c d e f g) 0) $\Longrightarrow$ ()
(**frente-lista** '() 3) $\Longrightarrow$ Error: la longitud de () es menor que 3

4. (**5 puntos**) Defina un procedimiento **n-tupla->entero** que convierta una tupla no vacía de dígitos a un número. Pruebe su procedimiento con los siguientes ejemplos:

   (**n-tupla->entero** '(3 1 4 6)) $\Longrightarrow$ 3146
   (**n-tupla->entero** '(0)) $\Longrightarrow$ 0
   (**n-tupla->entero** '()) $\Longrightarrow$ Error: el argumento () no es aceptable
   (+ (**n-tupla->entero** '(1 2 3)) (**n-tupla->entero** '(3 2 1))) $\Longrightarrow$ 444

5. (**Bonificación: 10 puntos**) De su curso de Cálculo sabe que:

$$\lim_{n\to\infty}\left(1+1+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{n!}\right)=e \tag{1}$$

   Defina un procedimiento llamado **calcula-e**, que tome como entrada un entero **n**, y regrese la suma de los primeros **n** términos de esta serie (infinita). El resultado será una aproximación de **e**. Pruebe su procedimiento con los siguientes ejemplos:

   (**calcula-e** 10) $\Longrightarrow$ 9864101/3628800
   (**calcula-e** 20) $\Longrightarrow$ 6613313319248080001/2432902008176640000

   IMPORTANTE: No intente usar este procedimiento con un valor demasiado grande de **n**, pues podría ocasionar un desbordamiento de memoria en ciertas implementaciones de Scheme.

Le resultará útil leer el capítulo 3 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-tarea2.scm**) en un CD que se entregará junto con el resto de su tarea. Las soluciones deben implementarse con los nombres que se indican en cada problema de los antes enumerados. No olvide incluir el código de sus procedimientos en el reporte de su tarea.

Fecha de entrega: *Martes 2 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.