Programación Declarativa

Ingeniería Informática
Cuarto curso. Primer cuatrimestre

Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

Curso académico: 2017 - 2018

Práctica número 3.- Iteración, recursión y funciones usadas como parámetros o devueltas como resultados

1. **Números amigos y perfectos**
   - Dos números naturales son **amigos** si la suma de los divisores de uno es igual al otro número y viceversa.
   - El menor par de números amigos es el formado por el 220 y 284:
     - Suma de los divisores de 220 (excepto 220):
       1 + 2 + 4 + 5 + 10 + 20 + 11 + 22 + 44 + 55 + 110 = 284
     - Suma de los divisores de 284 (excepto 284):
       1 + 2 + 4 + 71 + 142 = 220
   - Otros pares de números amigos son 1184 y 1210, 6232 y 6368, 2620 y 2924,...

   a. Codifica una función <u>*iterativa*</u>, denominada **suma-divisores,** para calcular la suma de los divisores de un número natural (excepto el propio número).
   b. Utiliza la función **suma-divisores** para codificar un predicado, denominado **amigos?**, que permita comprobar si dos números son o no amigos.
   c. Utiliza el predicado **amigos?** para codificar un predicado, denominado **perfecto?,** que permita comprobar si un número es perfecto, es decir, es igual a la suma de sus divisores inferiores a él.
      - Por ejemplo: el número 28 es perfecto
        28 = 14 + 7 + 4 + 2 + 1

2. **Algoritmo de Euclides**
   - El algoritmo de ***Euclides*** permite calcular el máximo común divisor (M.C.D.) de dos números naturales.
     - Si *"a"* y *"b"* son dos números naturales y $a = c\ b + r$ entonces *M.C.D.(a, b) = M.C.D.(b, r)*.
     - El algoritmo concluirá cuando el segundo argumento sea <u>cero</u>, siendo el máximo común divisor el primer argumento.

   - Ejemplo: cálculo del máximo común divisor de 630 y 198

a	**630**	198	36	**18**
b	**198**	36	18	0
r	36	18	0	

   M.C.D.(630,198) = 18

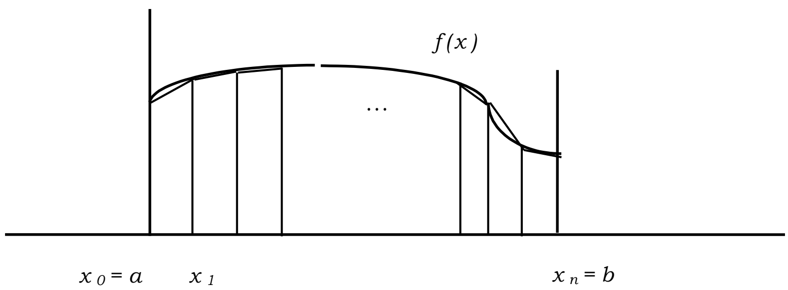
   a. Codifica una función <u>iterativa</u>, denominada **mcd-iterativo**, que permita calcular el máximo común divisor de dos números.
   b. Codifica una función <u>recursiva</u>, denominada **mcd-recursivo**, que permita calcular el máximo común divisor de dos números.

3. **Número primo**
   - Un número es **primo** si no tiene divisores propios <u>menores</u> que su raíz cuadrada.
     a. Codifica un predicado <u>iterativo</u>, denominado **primo-iterativo?**, para comprobar si un número es primo o no.
     b. Codifica un predicado <u>recursivo</u>, denominado **primo- recursivo?**, para comprobar si un número es primo o no.

4. **Integral definida**
   a. Codifica una función <u>iterativa</u>, denominada **integral**, que
      - reciba cuatro parámetros:
        - Los dos extremos de un intervalo: *a* y *b*
        - Una función que sea positiva en el intervalo *[a,b]: f*
        - Un número: *n*
      - y devuelva la aproximación a la integral definida según el **método de los trapecios**.

$$\int_a^b f(x) \quad dx = \sum_{i=0}^{n-1} \left(\frac{f(x_i) + f(x_{i+1})}{2} \right) * h$$

donde $h = (b - a) / n$ y $x_i = a + i * h$



   b. ¿Cómo se calcularía el área de la función $f(x) = 3 x^2 + 1$ definida en el intervalo [0,2]?

5. **Suma de series convergentes basadas en una cota de error**
   a. Codifica una función <u>iterativa</u> que permita calcular la suma de cualquier serie numérica convergente teniendo en cuenta una <u>cota de error</u>.

$$serie = \sum_{\substack{n = inicial \\ n = n + siguiente \ (n)}} f(n)$$

      - La función recibirá como parámetros
        - Una función que represente el término general de la serie: *f*
        - El índice del primer término: *inicial*
        - Una función que permita pasar al siguiente término de la serie: *siguiente*
        - Una cota de error de forma que la suma de la serie <u>finalizará</u> cuando el valor absoluto del término actual que se vaya a sumar sea menor que dicha cota de error: $|f(n)| < cota$
   b. Codifica una versión <u>recursiva</u> de la función anterior.
   c. Utiliza las funciones anteriores para comprobar que la siguiente serie numérica propuesta por **Leibniz** permite calcular una aproximación a $\pi/4$:

$$\frac{\pi}{4} = \sum_{n=1}^{\infty} f(n) = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots$$

6. **Número e**
   - Considérese el término general de una sucesión numérica que converge al número **e**: 2.718281…

$$a_n = (1 + 1/n)^n$$

   - Codifíquense las siguientes funciones:
     - **término-número-e**
       - Calculará el término n-ésimo de la sucesión numérica.
       - Recibirá como parámetro el valor de ***n*.**
     - **límite-sucesión-número-e-iterativa**
       - Se debe codificar una función iterativa que permita calcular el límite de la sucesión numérica que converge al número *e*.
       - La función debe recibir como argumento la **cota de error**, que permitirá terminar la función cuando dos elementos consecutivos de la sucesión disten menos que dicha cota de error: $|a_{n+1} - a_n| < cota$

7. **Límite de cualquier sucesión numérica convergente**
   a. Codifica una función iterativa denominada **"límite-iterativa"** que permita calcular una aproximación al límite de cualquier sucesión numérica convergente.
      - La función debe recibir como argumentos a:
        - Una **función** que represente el término general de la sucesión numérica convergente.
        - La **cota de error**, que permitirá terminar la función cuando dos elementos consecutivos de la sucesión disten menos que dicha cota de error.
   b. ¿Cómo se llamaría a la función "límite-iterativa" si se desea calcular el límite de la sucesión numérica cuyo término general es $a_n = (1 + 1/n)^n$ con una cota de error de 0.001?

8. **El número áureo**
   - El número áureo se define como

$$\varphi = \frac{1 + \sqrt{5}}{2} = 1{,}61803398 \quad 9.....$$

   - El número áureo también se puede calcular mediante la siguiente suma infinita

$$\varphi = \sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + \sqrt{1 + \ldots\ldots}}}}}}$$

   a. Codifica una función iterativa denominada **"suma-aureo-iterativo"** que permita calcular el número áureo usando la suma anterior. La función recibirá como parámetro el número de sumandos.
   b. Codifica una función recursiva denominada "**suma-aureo-recursivo"** que permita calcular el número áureo usando la suma anterior. La función recibirá como parámetro el número de sumandos.

9. **Fracción continua**
   - Codifica una función **iterativa** que permita calcular una aproximación a π usando la siguiente fracción continua:

$$\pi = \cfrac{4}{1 + \cfrac{1}{3 + \cfrac{4}{5 + \cfrac{9}{7 + \cdots}}}}$$

   - La función recibirá como parámetro el número de fracciones continuas que debe calcular.

10. **Serie de productos o “productorio”**
   - **Wallis** propuso utilizar la siguiente serie para calcular una aproximación a $\pi/4$:

$$\frac{\pi}{4} = \frac{2 \times 4 \times 4 \times 6 \times 6 \times 8 \times 8 \times \ldots}{3 \times 3 \times 5 \times 5 \times 7 \times 7 \; v \; 9 \times \ldots}$$

   a. Codifica una función denominada **factor-Wallis** que reciba como parámetro un número natural **n** y devuelva como resultado el n-ésimo factor de la sucesión de Wallis.
      - Por ejemplo:

        (factor-Wallis 1) ==> *2/3*
        (factor-Wallis 2) ==> *4/3*
        ...
        (factor-Wallis 5) ==> *6 /7*

   b. Escribe una función iterativa denominada **Wallis-iterativa** que reciba como parámetro un número natural que indicará cuántos factores se han de multiplicar.
   c. Escribe función recursiva de cola denominada **Wallis-recursiva** que reciba como parámetro una cota de error, de forma que la función terminará su ejecución cuando se verifique la siguiente desigualdad:

      1 - cota < factor < 1 + cota

   - **Observación:** la sucesión de Wallis converge “muy lentamente”.

Funciones que devuelve una función

11. Codifica una función denominada ***incremento-funcional*** que reciba una función ***f*** como parámetro y devuelva como resultado la función que calcularía la siguiente expresión

$$\frac{f(x+1) - 2f(x) + f(x-1)}{4}$$

   - ¿Cómo se invocaría la función ***incremento-funcional***? Pon un ejemplo.

12. Codifica una función denominada ***diferencia-simétrica*** que reciba como parámetros dos funciones ***f*** y ***g*** y devuelva como resultado la función que calcularía la siguiente expresión:

$$|f(x) - g(x)|$$

   - ¿Cómo se invocaría la función ***diferencia-simétrica***? Pon un ejemplo.