



PRÁCTICO N° 8

Introducción

El objetivo de este práctico es profundizar la comprensión de la metodología de programación llamada recursión, en primera instancia con ejercicios de sistema de numeración y luego con ejercicios de dificultad superior.

Ejercicio 1

Definimos *SumRestDig* de un número entero positivo al número que se obtiene de sumar sus dígitos de posición impar y restar sus dígitos de posición par (el dígito 1 es el que está más a la derecha).

Ej.: el número *SumRestDig* de 2694517 es 12 pues $+2-6+9-4+5-1+7 = 12$

Escriba una función recursiva para obtener el número *SumRestDig* de un entero positivo dado.

Ejercicio 2

Implemente una función recursiva que reciba un valor entero (decimal) y devuelva su equivalente en sistema binario en un vector.

Ejemplo: 47 en binario es 101111
paso 1: $47 / 2 = 23$ resto 1
paso 2: $23 / 2 = 11$ resto 1
paso 3: $11 / 2 = 5$ resto 1
paso 4: $5 / 2 = 2$ resto 1
paso 5: $2 / 2 = 1$ resto 0

Ejercicio 3

Implemente la función recursiva *BinarioADecimal* que tome un natural binario (≥ 0) representado como un vector de ceros y unos, y retorne el decimal correspondiente.

Ejercicio 4

Escriba una función recursiva *abase10* que reciba como parámetros un número entero positivo N expresado en base b y la base b , y devuelva su equivalente en base 10. El número N estará representado por un vector, donde cada elemento representa un dígito de N en base b .

Ejemplos:

```
>>y= abase10([1, 0, 1, 0, 0], 2)
y = 20
>>y= abase10([7, 4, 3], 8)
y = 483
```

Ejercicio 5

Escriba una función recursiva que determine la cantidad de celdas de valor 1, en una matriz M dada de números enteros, cuadrada de dimensión $2^n \times 2^n$ (donde n es un número natural mayor que 0, pasado como parámetro).



Ejercicio 6

Implementar en Octave la función recursiva *IndMayorRec* la cual recibe como parámetro un vector y devuelve el índice donde se encuentra el máximo. Si se encuentra más de una vez, tomar el de menor índice.

- Realice la función con una estrategia de recursión desde el último lugar del vector hacia el primer lugar del vector.
- Realice la función con una estrategia de recursión desde el primer lugar del vector hacia el último lugar del vector.

Ejercicio 7

Los polinomios de Hermite se definen por las siguientes fórmulas:

$$H_0(x) = 1$$

$$H_1(x) = 2x$$

$$H_n(x) = 2xH_{n-1}(x) - 2(n-1)H_{n-2}(x), \text{ para } n > 1$$

- Escriba una función recursiva $h = \text{hermite}(n,x)$ que calcule el valor del polinomio de Hermite H_n de grado n en el punto x .
- Escriba una función iterativa $h = \text{hermitit}(n,x)$ que calcule lo mismo.
Sugerencia: utilice una estructura de iteración: **for** entre $i = 2$ e $i = n$, y mantenga en dos variables los valores de $H_{i-1}(x)$ y $H_{i-2}(x)$.
- ¿Cuál de las dos es más eficiente y por qué? Para averiguarlo haga lo siguiente, agregue al comienzo del cuerpo de *hermite* una línea como la siguiente:

disp(['Comienzo de hermite(' num2str(n) ', x')]);

También agregue una línea similar al final del cuerpo de la misma.

¿Cuántas veces es llamada recursivamente la función *hermite*? Comparar con el número de veces que se ejecuta el ciclo en *hermitit*.

Ejercicio 8

Suponiendo que una función $f(x)$ tiene una raíz única en el intervalo $[a,b]$, se desea implementar en Octave el *método de bisección* para calcular la raíz de $f(x)$ en $[a,b]$.

El método de bisección se basa en el Teorema de Bolzano, que demuestra que si $f(a)$ y $f(b)$ tienen signos distintos entonces f debe tener, al menos, una raíz en el intervalo $[a,b]$. En cada paso, el *método de bisección* divide el intervalo en dos, usando un tercer punto $c=(a+b)/2$. En ese momento existen dos posibilidades: o bien $f(a)$ y $f(c)$ tienen signo distinto, o bien $f(c)$ y $f(b)$ tienen signo distinto. A continuación, el método selecciona el intervalo adecuado y continúa buscando la raíz, hasta que el tamaño del intervalo sea menor que cierto valor de tolerancia.

- Escribir una función recursiva *bisec_rec* que reciba la función a evaluar, los extremos a y b y el valor de tolerancia tol , y que devuelva la raíz de f en $[a,b]$ calculada a partir del *método de bisección*.
- Escribir una función iterativa *bisec_iter*, análoga a la anterior, pero implementada como un algoritmo iterativo.

Ejemplo de cabecal de la función: `function root=bisec_rec(funcion, a, b, tol)`

Ejemplo de uso de la función en Octave: `raiz= bisec_rec(@cos, 0, 3.14, 0.0001)`, donde $a = 0$, $b = 3.14$ y $tol = 0.0001$

Nota: Se recomienda utilizar las funciones *feval* y *sign* en la implementación de los ejercicios.



Ejercicio 9

Escriba una función recursiva *Union* donde a partir de dos vectores, u y v , que tienen sus elementos ordenados de forma ascendente y sin elementos repetidos, devuelva un vector, w , que contiene todos los elementos de u y v ordenados de forma ascendente y sin repeticiones.

Ejercicio 10

Escriba en Octave la función recursiva *masParesRec*, que dado un vector v , devuelva true si el vector tiene más elementos pares que impares, y false en caso contrario.

Sugerencia: Utilizar una función cabezal para resolver el problema.

Ejercicio 11

El cálculo de raíz cuadrada puede realizarse de forma aproximada mediante un método iterativo.

Se estima inicialmente la solución como $s_1 = \frac{x}{2}$ lo cual sólo es cierto cuando $x = 4$. Seguidamente calculamos una nueva aproximación $s_{i+1} = s_i - \frac{s_i^2 - x}{2s_i}$.

Sea *tol* un valor arbitrario de tolerancia y cuando la diferencia en valor absoluto entre dos aproximaciones sucesivas satisface que $|s_{i+1} - s_i| < tol$, se considerará que la última aproximación es una buena aproximación al valor de la raíz cuadrada. En caso contrario, se realiza una nueva aproximación.

Escriba en Octave la función recursiva *calcularRaiz_rec* que reciba como parámetros de entrada dos números reales positivos (el valor del que se va a calcular la raíz cuadrada y el valor *tol*) y retorne como parámetro de salida el valor calculado para la raíz cuadrada. Para poner a prueba su función, utilice valores pequeños de *tol*; por ejemplo: $tol = 0.0001$.

Sugerencia: Utilizar una función cabezal para resolver el problema.

Ejercicio 12

- a) Considere una función $f: \mathbf{N} \rightarrow \mathbf{R}$. Podemos representar los n primeros valores de dicha función mediante un vector en Octave $[f_1 f_2 \dots f_n]$.

Se denominará *derivada discreta de f en i* (se nota como Df_i) al valor $f_{i+1} - f_i$

Escriba una función recursiva “**derivadadiscrec**” en Octave que tome como entrada un vector que contenga los n primeros valores de f y devuelva un vector que represente los $n-1$ primeros valores de la *derivada discreta de f*.

- b) Dado un vector Df que representa la *derivada discreta de f* y una constante C , es posible hallar la *integral discreta de Df en i* (se nota como S_i) mediante la recurrencia:

$$S_1 = C$$

$$S_{i+1} = S_i + Df_i, \text{ para } i \geq 1$$

Observar que realizando una elección apropiada de la constante C (específicamente tomando $C=f$), la *integral discreta* de todos los elementos del vector Df dará como resultado el vector original con los n primeros elementos de f .

Escriba una función recursiva “**integraldiscrec**” en Octave que tome como entrada un vector Df y una constante C y devuelva un vector con los valores $[S_1 S_2 \dots S_n]$.