
Programación

Recursividad

- - Ejercicios -

EJERCICIOS

Recursividad

Ejercicio 1

El siguiente ejercicio es el cálculo del factorial de un número en Java. Se pide implementar una función similar en Python.

```
public static int factorial(int n){  
  
    // Si n = 0 entonces  
    //          0! = 1  
    // si n > 0 entonces  
    //          n! = n * (n-1)! = n * (n-1) * (n-2) * ... * 3 * 2 * 1  
  
    if (n == 0){  
        return 1;  
    }  
    else{  
        return n * factorial(n - 1);  
    }  
}
```

Ejercicio 2

a.- Escribir un método de clase recursivo que muestre en pantalla los números naturales **del 1 al n**, donde $n > 0$ es el valor pasado como parámetro en la llamada inicial.

b.- Escribir un método de clase recursivo que muestre en pantalla los números naturales **del n al 1**, donde $n > 0$ es el valor pasado como parámetro en la llamada inicial.

Ejercicio 3

Escribir qué se muestra por pantalla al ejecutar este programa:

NO ENTREGAR

```
public static void escribeRaro(int n) {  
  
    if (n>0) {  
        System.out.print(n);  
        escribeRaro(n-1);  
        System.out.print(n);  
    }  
    else {  
        System.out.print(0);  
    }  
}  
  
public static void main(String[] args) {  
  
    escribeRaro(5);  
}
```

Ejercicio 4

Escribir un método de clase recursivo que, dados 2 números naturales $a \geq 0$ y $b > 0$, calcule el cociente de su **división entera**, basándose en el hecho de que dicha operación se puede realizar como una serie de restas sucesivas, siguiendo la recurrencia:

$$a/b = 0, \text{ si } a < b,$$

$$a/b = (a-b)/b + 1, \text{ si } a \geq b.$$

Ejercicio 5

Dados dos números enteros a y b , siendo $b \geq 0$, se puede definir recursivamente su **producto** $a \cdot b$ del modo siguiente:

$$a \cdot b = 0, \text{ si } b = 0,$$

$$a \cdot b = a \cdot (b-1) + a, \text{ si } b > 0.$$

Nótese que, para esta definición, la multiplicación se reduce a una secuencia de sumas. Considerando que tan sólo es posible utilizar operaciones aditivas, escribir un método de clase recursivo para realizar la operación pedida, siguiendo la definición anterior.

Definición:

Recursividad: -ver Recursividad.

Acrónimos recursivos:

GNU: GNU's Not Unix

PHP: PHP Hypertext Preprocessor

WINE: WINE Is Not an Emulator



Ejercicio 6

Dados dos números enteros a y b , siendo $b \geq 0$, otra forma de definir recursivamente su **producto** $a \cdot b$ es la siguiente:

$$a \cdot b = 0, \text{ si } b = 0,$$

$$a \cdot b = (a \cdot 2) \cdot (b/2), \text{ si } b > 0 \text{ y } b \text{ es par, y}$$

$$a \cdot b = (a \cdot 2) \cdot (b/2) + a, \text{ si } b > 0 \text{ y } b \text{ es impar.}$$

Este tipo de multiplicación se conoce como **multiplicación a la rusa**, siendo utilizada antiguamente por los comerciantes de dicho país, que no conocían la tabla de multiplicar, para efectuar el producto de dos números positivos cualesquiera utilizando solamente sumas, productos y divisiones por 2.

Considerando que tan sólo se pueden utilizar productos y divisiones por 2, así como sumas, escribir un método de clase recursivo para realizar la operación pedida, siguiendo la definición anterior.

Ejercicio 7

Dados dos números enteros $a \geq 0$ y $b \geq 0$, se puede definir recursivamente la **potencia** a^b del modo siguiente:

$$a^b = 1, \text{ si } b = 0,$$

$$a^b = a, \text{ si } b = 1,$$

$$a^b = (a^{b/2}) \cdot (a^{b/2}), \text{ si } b > 1 \text{ y } b \text{ es par, y}$$

$$a^b = (a^{b/2}) \cdot (a^{b/2}) \cdot a, \text{ si } b > 1 \text{ y } b \text{ es impar.}$$

Considerando que tan sólo se pueden utilizar divisiones por 2 y productos, escribir un método de clase recursivo para realizar la operación pedida, siguiendo la definición anterior.

Ejercicio 8

- a) Escribir un método de clase recursivo que devuelva la suma de los dígitos de un número natural $n \geq 0$ pasado como parámetro.
 - b) Escribir un método de clase recursivo que devuelva el número de dígitos de un número natural $n \geq 0$ pasado como parámetro.
 - c) Escribir un método de clase recursivo que devuelva en orden inverso los dígitos que componen un número natural $n \geq 0$ dado.
-

Ejercicio 9

Escribir un método de clase recursivo que devuelva el valor binario de un número natural $n \geq 0$ dado.

Por ejemplo:

si $n=5$ el método devuelve 101

si $n=31$ el método devuelve 11111

$$\begin{array}{c} 100011_{(2)} = 35_{(10)} \\ \swarrow \quad \downarrow \quad \searrow \quad \swarrow \quad \searrow \quad \swarrow \\ 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ 32 + 0 + 0 + 0 + 2 + 1 = 35_{(10)} \end{array}$$

$$\begin{array}{r} 153 \div 2 = 76 \text{ r } 1 \\ 76 \div 2 = 38 \text{ r } 0 \\ 38 \div 2 = 19 \text{ r } 0 \\ 19 \div 2 = 9 \text{ r } 1 \\ 9 \div 2 = 4 \text{ r } 1 \\ 4 \div 2 = 2 \text{ r } 0 \\ 2 \div 2 = 1 \text{ r } 0 \\ 1 \div 2 = 0 \text{ r } 1 \end{array} \quad 153_{10} = 10011001_2$$

EJERCICIOS

Recursividad & Vectores

Ejercicio 10

Dado un array de enteros v , escribir un **método de clase recursivo** que:

- a) Imprima todos los elementos del vector de forma Ascendente.
 - b) Imprima todos los elementos del vector de forma Descendente.
 - c) Obtenga la suma de todos los elementos del array.
 - d) Dado un entero x , cuente cuántas veces aparece en el array.
 - e) Compruebe si el array está ordenado ascendentemente.
 - f) Obtenga la posición en el vector de un valor pasado como parámetro.
 - g) Dadas dos posiciones, izq y der , del array, $0 \leq izq \leq der \leq v.length-1$, duplique el valor de los elementos del array situados entre dichas posiciones.
 - h) Obtenga el valor máximo (mínimo) del array.
 - i) Obtenga la posición del máximo (mínimo) del array.
 - j) Determine la posición del primer (último) elemento no nulo del array.
-

```
if( !empty( stomach ) ){  
    keepCoding();  
}  
else{  
    orderPizza();  
}
```



Ejercicio 12

- k) Determine cuántos ceros consecutivos hay al final del array.
 - l) Dadas dos posiciones, *izq* y *der*, del array, $0 \leq \text{izq} \leq \text{der} \leq \text{v.length}-1$, invierta todos los elementos del array situados entre dichas posiciones, esto es, al finalizar la ejecución del método el array contendrá en su posición *izq* el elemento que inicialmente ocupaba la posición *der*, en su posición *izq*+1 el elemento que inicialmente ocupaba la posición *der*-1 y así sucesivamente.
 - m) Dado un entero $b > 0$, determine si *b* es igual a la suma de todas las componentes de *v*.
 - n) Dado un entero *x*, determine la cantidad de elementos del array que son menores que *x*.
 - o) Determine la cantidad de elementos impares que ocupan posiciones pares del array.
 - p) Determine la posición, si existe, de la primera subsecuencia del array que comprenda, al menos tres números enteros consecutivos en posiciones consecutivas del array.
-

Ejercicio 13

Escribir un método de clase recursivo que, dados dos String `s1` y `s2` y sin hacer uso de los métodos definidos en la clase String que resuelven el mismo problema, determine:

- a) si `s2` es prefijo de `s1`.
- b) si `s2` es sufijo de `s1`.
- c) si `s2` es una subcadena de `s1`.

Ejercicio 14

- a) Escribir un método de clase recursivo que, dados un String `s` y su longitud `l`, muestre en **orden inverso** los caracteres de `s`.

- b) Escribir un método de clase recursivo que dé el mismo resultado que el apartado anterior pero recibiendo sólo la cadena (String). Ayuda: usar el método `substring`.

Ejercicio 15

Escribir un método de clase recursivo que compruebe si un String `s` dado es **palíndromo**.

Ayuda: usar el método `substring` para reducir la longitud de `s` en cada llamada recursiva.



Ejercicio 16

Dado un array de String `v`, escribir un método de clase recursivo que:

a) determine si es **capicúa**, esto es, si la primera y última palabra del array son la misma, la segunda y la penúltima palabras también lo son, y así sucesivamente. El método retornará `true` si el array es capicúa o `false` en caso contrario.

b) dada una palabra `pal`, determine **la existencia** de dicha palabra en el array entre dos posiciones dadas `ini` y `fin` que cumplen inicialmente: $0 \leq ini \leq fin < v.length$. Caso de existir la palabra, el método devolverá la primera posición donde se encuentre la misma, y de no existir el método devolverá el valor `-1`.