**RECURSIVIDAD**

Solución de programación de cálculos por RECURRENCIA.

**RECURRENCIA (recurrency)**

La RELACIÓN DE RECURRENCIA, para cálculos, es una secuencia de términos, ecuaciones, de valores, u objetos, que basa la definición de cada siguiente nuevo término en función de términos ya definidos.

RECURRENCIA por

* ITERACIÓN (iteration), CICLOS, o REPETICIÓN (for, while, repeat, do); y
* RECURSIVIDAD (recursión)

**ITERACIÓN (iteration)**

Recurrencia por ciclos o repeticiones usando estructuras algorítmicas repetitivas (for, while, …)

|  |  |  |  |
| --- | --- | --- | --- |
| Sumatoria | Factorial | m \* n | etiquetas |
| suma=0;  suma = suma + A[0];  suma = suma + A[1];  . . .  suma = suma + A[n-1];  suma = suma + A[n]; | fact=1;  fact= 1\*fact;  fact= 2\*fact;  . . .  fact= (n-1)\*fact;  fact= n\*fact; | mul = 0;  mul = mul + m;  mul = mul + m;  . . .  mul = mul + m;  mul = mul + m; | SALUDOS::1  SALUDOS::2  . . .  SALUDOS::n-1  SALUDOS::n |

Dentro de los algoritmos que han resuelto de esta manera están:

* una sumatoria,
* el factorial,
* la multiplicación por medio de sumas,
* impresión de etiquetas,
* el cálculo de la potencia,
* la serie de Fibbonacci,
* la búsqueda secuencial de un valor
* obtención del mayor valor de un vector
* la búsqueda binaria,
* el ordenamiento de un vector por selección directa;
* y muchos más.

Por ejemplo, programa en Java para imprimir etiquetas empleando recurrencia muy simple:

// Imprimir n etiquetas iterativamente.

public static void main(String args[ ])

{

// Imprimir n etiquetas recurrentemente.

int i;

for(i=1; i<=4; i=i+1) {

System.out.println(“SALUDOS” + "::" + i);

}

}

O mediante un método

public static void main(String args[ ])

{

// Imprimir n etiquetas iterativamente.

**impNetiq1**(“SALUDOS”, 4);

}

// Imprimir n etiquetas iterativamente.

public static void **impNetiq1**(String Etiqueta, int n) {

// Imprimir n etiquetas iterativamente.

int i;

for(i=1; i<=n; i=i+1) {

System.out.println(Etiqueta + "::" + i);

}

}

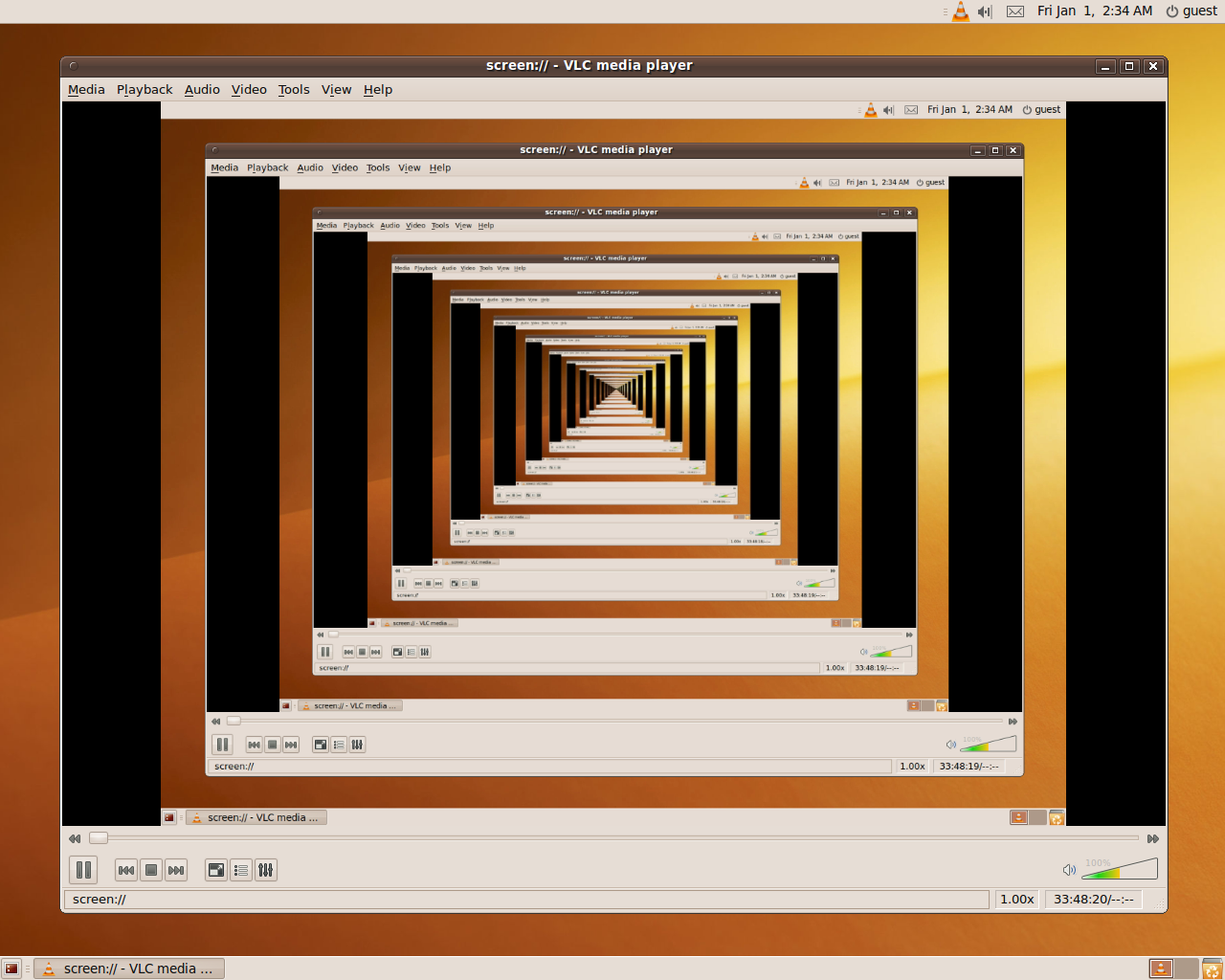
**RECURSIVIDAD**

Recursividad, en matemáticas o en las ciencias de la computación, es un método donde el término que está siendo definido es aplicado dentro de su propia definición.

Este método también se aplica en el proceso de objetos que se definen parcialmente a partir de sí mismos, por ejemplo:



Un forma visual de recursividad. En ésta imagen (anuncio) se muestra a una mujer que sostiene un objeto (caja sobre charola) que contiene una imagen más pequeña de ella sosteniendo el mismo objeto (caja sobre charola), el cual a su vez contiene una imagen aun más pequeña de ella sosteniendo el mismo objeto (caja sobre charola), y así repetidamente.



Recursion in a screen recording program, where the smaller window contains a snapshot of the entire screen.

Cuando las superficies de dos espejos están exactamente en paralelo, una con respecto a la otra, las imágenes anidadas que se forman son una forma de una recursividad infinita.

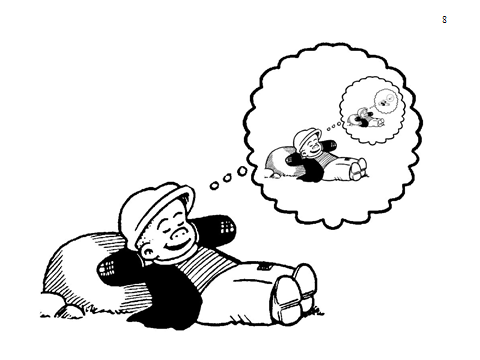


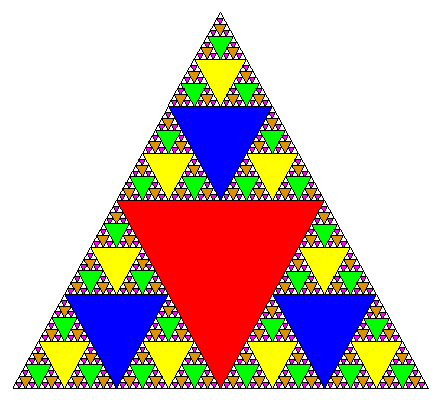
This screenshot of a web page includes the screen shot itself.

A **Russian nesting doll** or a **matryoshka doll (taken apart, set in a row)**







A Sierpinski triangle—a confined recursion of triangles to form a geometric lattice

En el lenguaje.

Un REY es hijo de un REY.

Un REY es hijo de un REY.

Un REY es hijo de un REY.

: : :

Un REY es hijo de un CONQUISTADOR.

**RECURSIVIDAD en métodos (o funciones), excepto *main( )***

Si hiciéramos un símil entre una imagen que se muestra y una rutina en funcionamiento, diríamos que la rutina como una de sus actividades, tiene que llamarse (invocarse) a si misma:

public static void rutinaX(…) // Como si fuera un imagen recursiva

{

int a, …

…

… rutinaX(…) // llamado o invocación así misma

…

}

Dentro de una rutina recursiva no hay instrucciones algorítmicas repetitivas como FOR, WHILE, REPEAT, DO, etc; pero si se pueden usar IF y SWITCH.

El método (o función) puede ser estático o de instancia.



Una método (objeto, término, ecuación) exhibe recursividad cuando puede ser definido con dos propiedades (excluyentes entre sí):

* la de uno o varios casos base (no recursivo(s)),
* la de uno o varios caso(s) recursivo(s).

El caso o casos base, son los términos conocidos y es donde deben detenerse las definiciones.

Ejemplo del factorial de *n*:

* fact(0)=1; fact(1)=1; (casos base), y
* fact(2)=2\*fact(1); fact(3)=3\*fact(2); … fact(n)=n\*fact(n-1); (casos recursivos).

**Llamada a un método recursivo:**

En el caso recursivo, cada llamada genera una nueva copia de los parámetros formales y las variables locales.

El método se ejecutará normalmente hasta la llamada a sí misma. En ese momento se crean en la pila nuevos parámetros formales y variables locales.

El nuevo ejemplar del método comienza a ejecutarse.

Se crean más copias hasta llegar a los casos bases, donde se resuelve directamente el valor, y se va saliendo liberando memoria hasta llegar a la primera llamada (última en cerrarse)