GracoSoft

Unidad Didáctica II
RECURSIVIDAD

Contenido y Diseño: Instructor Carmelo Yonso
Fecha: Abril 2023

GracoSoft

RECURSIVIDAD

Objetivos:

- Conocer los fundamentos y características de los algoritmos recursivos.
- Conocer el funcionamiento de las funciones recursivas.
- Conocer las características fundamentales de las estructuras de datos recursivas.
- Conocer las ventajas y desventajas de las funciones recursivas frente a las funciones iterativas.

GracoSoft

RECURSIVIDAD

Introducción:

- Una técnica común de resolución de problemas es la división de un problema en varios subproblemas de la misma categoría, pero de más fácil resolución. Esta técnica se conoce como divide y vencerás.
- Un ejemplo de algoritmos en los que se aplica esta técnica son los algoritmos recursivos, en los cuales el algoritmo se llama a sí mismo repetidamente, procurando simplificar o reducir el problema en cada llamada, hasta llegar a un caso trivial de solución directa.

GracoSoft

RECURSIVIDAD

Definición:

- Se dice que un algoritmo es recursivo si dentro del cuerpo del algoritmo y de forma directa o indirecta se realiza **una llamada a él mismo**.
- Al escribir un algoritmo recursivo, debe establecerse de algún modo cuando debe dejar de llamarse a sí mismo, o de otra forma se repetiría indefinidamente. Para ello, se establece una condición de salida llamada **caso base**.
- Se llama caso base o condición de salida al caso trivial de un algoritmo recursivo, del cual conocemos su solución.
- El caso base contempla el caso en el cual la solución es lo suficientemente sencilla como para responder directamente sin necesidad de realizar otra llamada al algoritmo.

```
1 reference
public int EjemploRecursividad(int numero)
{
    if (numero == 0) return 1;
    return numero * EjemploRecursividad(numero - 1);

}
```

RECURSIVIDAD

Ejemplo:

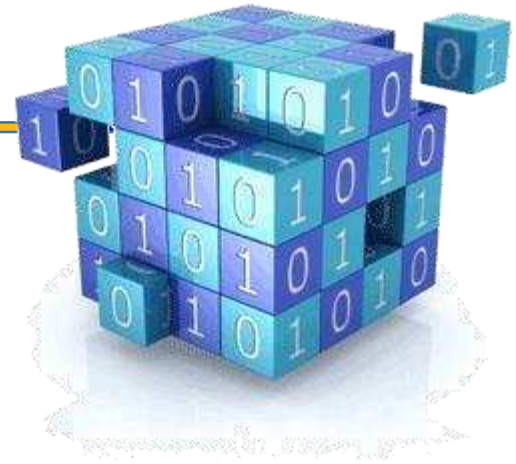
- El ejemplo más típico de algoritmo recursivo es el de una función para calcular el factorial de un número. El factorial de un número es el resultado de multiplicar dicho número por todos los precedentes, hasta llegar a 1.

  Por ejemplo, factorial(3) = 3 * 2 * 1.

- Si observamos que el factorial de un número es equivalente al producto de dicho número por el factorial del número precedente: factorial(3) = 3 * factorial(2) podemos plantear una implementación recursiva:

```
función factorial(n)
        si n = 1
                devolver 1
        sino
                devolver n * factorial(n – 1)
        fin si
fin función
```

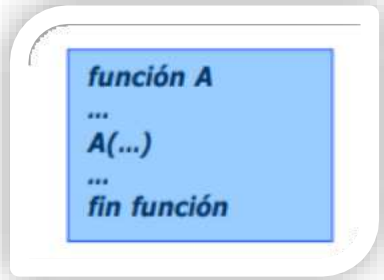
- En este caso, la sentencia si n = 1 devolver 1 es la condición de salida o caso base que evita que la función se llame a sí misma indefinidamente.

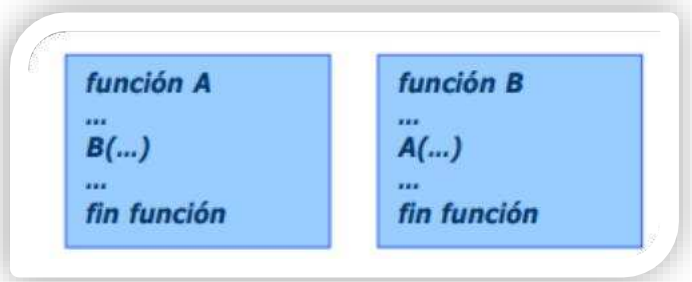


RECURSIVIDAD

Tipos:

- Según el punto desde el cual se hace la llamada recursiva: recursividad **directa** o **indirecta**.
  - **Directa.** Se da cuando la función efectúa una llamada a sí misma.
  - **Indirecta**. Se da cuando una función A llama a otra función B la cual a su vez, y de forma directa o indirecta, llama nuevamente a A.





RECURSIVIDAD

Tipos:

- Según el número de llamadas recursivas efectuadas en tiempo de ejecución: recursividad lineal o **no lineal**.
  - **Lineal o Simple.** Se da cuando la recursividad es directa y además cada llamada a la función recursiva sólo hace una nueva llamada recursiva.
  - **No lineal o Múltiple.** La ejecución de una llamada recursiva da lugar a más de una llamada a la función recursiva.

```
función A
...
A(...)
...
fin función
```

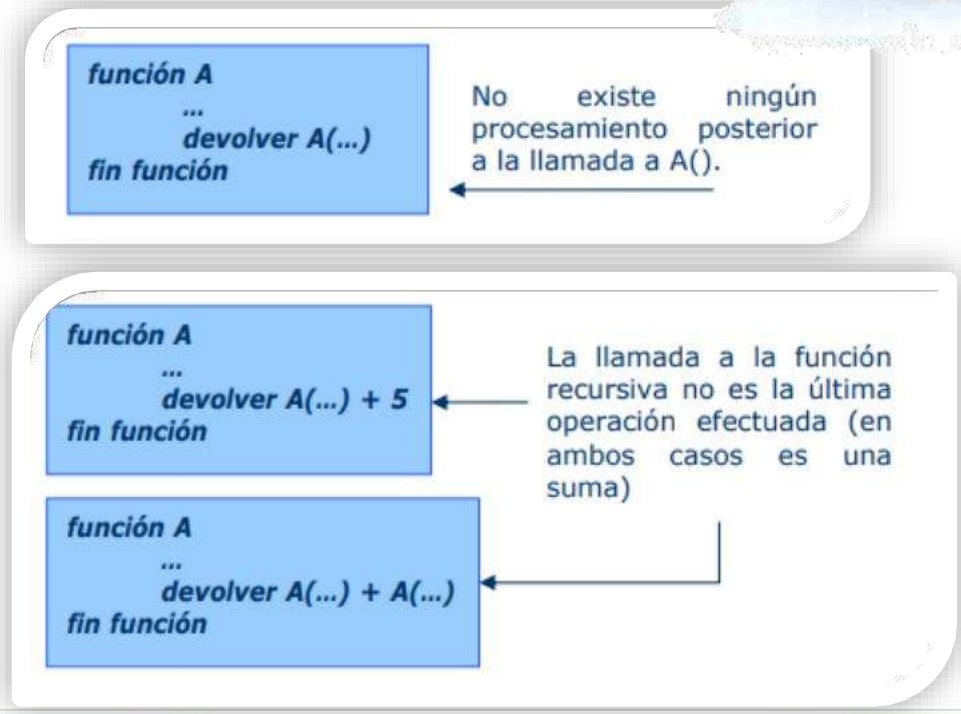
```
función A
...
si condición
        A(...)
fin si
...
A(...)
...
fin función
```

La recursividad será no lineal si se cumple ***condición***, pues en tal caso se producen dos llamadas recursivas.

RECURSIVIDAD

Tipos:

- Según el punto del algoritmo desde donde se efectúa la llamada recursiva: recursividad final o no final.
  - **Final.** Se da cuando la llamada recursiva es la última operación efectuada en el cuerpo de la función. (sin tener en cuenta la sentencia devolver)
  - **No Final.** Se da cuando la llamada recursiva no es la última operación realizada dentro de la función (sin tener en cuenta la sentencia devolver)



RECURSIVIDAD

Características:

- Un algoritmo recursivo consta de una parte recursiva, otra iterativa o no recursiva y una condición de terminación. La parte recursiva y la condición de terminación siempre existen.
- En cambio la parte no recursiva puede coincidir con la condición de terminación. Algo muy importante a tener en cuenta cuando usemos la recursividad es que es necesario asegurarnos que llega un momento en que no hacemos más llamadas recursivas. Si no se cumple esta condición el programa no parará nunca.

RECURSIVIDAD

Ventajas e Inconvenientes:

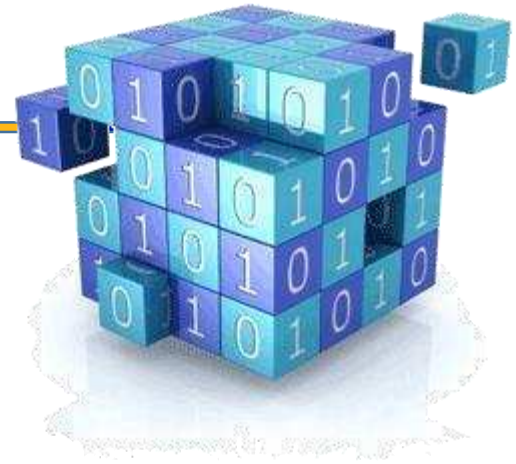
- La principal ventaja es la simplicidad de comprensión y su gran potencia, favoreciendo la resolución de problemas de manera natural, sencilla y elegante; y facilidad para comprobar y convencerse de que la solución del problema es correcta.
- El principal inconveniente es la ineficiencia tanto en tiempo como en memoria, dado que para permitir su uso es necesario transformar el programa recursivo en otro iterativo, que utiliza bucles y pilas para almacenar las variables.

RECURSIVIDAD

Funciones Recursivas vs Iterativas

- Se basan en una estructura de control: las iterativas utilizan una estructura de repetición; las recursivas una estructura de selección.
- Incluyen un ciclo de repetición: la iteración utiliza una estructura de repetición explícita; la recursión lo hace mediante llamadas de función repetidas.
- Incluyen una prueba de terminación: la iteración termina cuando falla la condición de continuación del ciclo; la recursión termina cuando se reconoce el caso base.
- Rendimiento: En el caso de la recursión el invocar repetidamente la misma función, puede resultar costosa en tiempo de procesador y espacio de memoria. Por el contrario la iteración se produce dentro de una función.

GracoSoft



RECURSIVIDAD

Escriba las siguientes funciones recursivas:

- Encuentre el estudiante que tiene el mayor promedio.
- La suma de las edades de los estudiantes.
- Busque un estudiante en el grupo y devuelva su posición en el arreglo.

GracoSoft

REFERENCIAS BIBLIOGRÁFICAS

Joyanes Aguilar, Luis (1996) Fundamentos de programación, Algoritmos y Estructura de datos. McGraw-Hill, México.

Deitel & Deitel (2001) C++ Como programar en C/C++. Prentice Hall

Kerrighan y Ritchie “El lenguaje de programación”. Prentice Hall

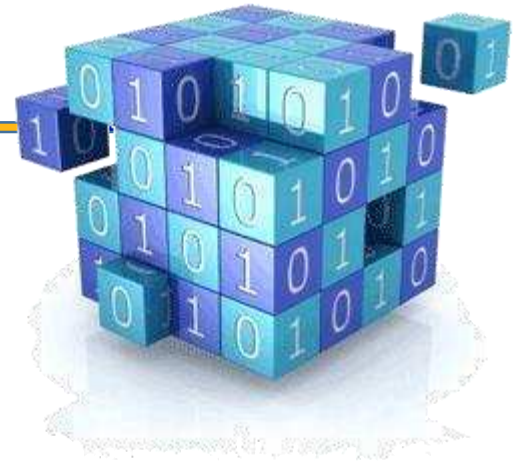
Gottfried, Byron (1999) “Programación en C” McGrawHill, México.

Levine Gutierrez, Guillermo (1990) Introducción a la computación y a la programación estructurada. McGraw-Hill, México.

Levine Gutierrez, Guillermo (1990) Introducción a la computación y a la programación estructurada. McGraw-Hill, México.

H. Schildt, C++ from the Ground Up, McGraw-Hill, Berkeley, CA, 1998

Keller,,AL;Pohl,Ira. A Book on C. 3 ª edición. Edit.Benjamin umnings.1995



Fin de la Unidad Didáctica II