Asignatura Introducción a la Programación

Ingeniería Informática

Guía de Aprendizaje

Tema : Programación funcional.

Unidad didáctica: Funciones y métodos de clase recursivos

1. **Objetivos de la unidad didáctica:**

- Caracterizar funciones y métodos de clase recursivos.
- Identificar los principales elementos que componen un método o función recursivo.
- Diseñar e implementar funciones y métodos recursivos.

2. **Introducción a la unidad didáctica:**

En guías de aprendizajes anteriores se había abordado que la mayoría de los programas de cómputo que resuelven los problemas reales son mucho más extensos que los programas que hemos desarrollado hasta ahora. La experiencia ha demostrado que la mejor manera de desarrollar y mantener un programa extenso es construirlo a partir de pequeñas piezas sencillas, o **módulos**.

Se había definido que existen tres tipos de módulos en Java: métodos, clases y paquetes. Y que el módulo más pequeño de los tres anteriores era: los métodos.

Un método era un grupo de instrucciones para realizar un objetivo procedimiento bien definido encapsuladas bajo un identificador que podían o no recibir uno o varios parámetros para su realización e incluso devolver o retornar algún valor.

El grupo de instrucciones que componen un método no se van a ejecutar mientras no se invoque o llame al método desde el método principal del programa *main* o desde otro método. Pero un método podrá auto-invocarse o auto-llamarse el mismo. Los métodos que dentro de sus instrucciones tienen un llamada a así mismo son los llamados métodos o funciones recursiva. En esta guía de aprendizajes estaremos abordando sobre los mismos.

Veamos el siguiente ejemplo: Hagamos un método que sea capaz de calcular el factorial de un numero entero N el cual es pasado por parámetro. Como sabemos el factorial de N se representar !N que por definición es la multiplicación de todos los números enteros entre 1 y N incluyendo los extremos y que para N =0 el factorial es 1. Los primeros seis factoriales son:

!0 = 0

!1 = 1

!2 = 2 = 2 * !1

!3 = 6 = 3 * !2

!4 = 24 = 4 * !3

!5 = 120 = 5 * !4

Ya nos podemos percatar que el factorial de N se puede definir como la multiplicación de N por el factorial de N-1 excepto en los casos de N igual a 1 o 0 donde el valor es 1. Mas formalmente podemos plantear matemáticamente como

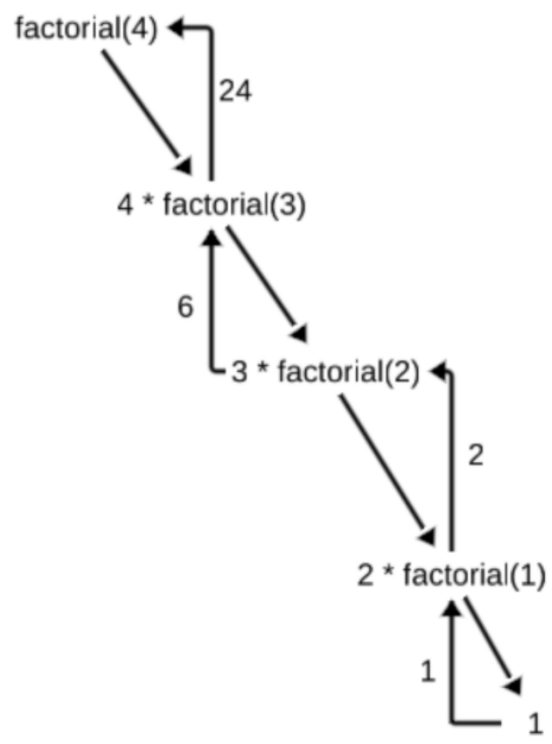
factorial(N) = 1 si N es 0 o 1

N * factorial(N-1) para cualquier N>1

Llevando lo anterior a lenguaje de programación de Java nos quedaría de la siguiente manera:

```
public static int factorial( int  n){
        if ( n==0 || n==1 ){
                return 1;
        } else {
                return n * factorial(n-1);
        }
}
```

Bueno si vemos la linea número 4 de método (return n * factorial (n-1)) se invoca o se llama al mismo método pero con la diferencia que el valor de N será pasado como parametro pero decrementado en uno. Si el método fue llamado con n = 4 en esa linea sería return 4 * factorial(3). Si traceamos el método recursivo y pintamos su árbol de traceo quedaría como se muestra en la siguiente imagen:



Como vemos en la imagen cuando factorial(4) invoca a factorial(3) este invoca a factorial(2)

y este a factorial(1) quien sencillamente retorna 1 que después es multiplicado por 2 (respuesta de la llamada a factorial (2)) que a su vez es multiplicado por 3 siendo 6 como resultado (resultado de factorial(3)) que es multiplicado por 4 que arroja 24 que es el resultado final de factorial(4) . Es evidente que los métodos recursivo desencadena una serie de llamadas en cadena hasta llegar a un punto que se retorna un valor y no a una invocación nuevamente a la función o método en si.

3. Orientaciones para el estudio:

1. Le sugerimos que para realizar el estudio de una materia debe elegir un lugar agradable, limpio, ventilado, cómodo, iluminado y si necesita recursos tecnológicos deben estar disponibles.

2. Cree hábitos de estudio sistemático; para esto:

    - Planifique su tiempo y no deje para mañana lo que pueda hacer hoy; una buena planificación hace manejable las responsabilidades diarias que tenemos para con la vida, permite cumplir con todas las tareas programadas y así evita agobios de última hora.

    - Calendarice las fechas más importantes de estudio y entrega de tareas. Localice con antelación los materiales que necesita para realizar el estudio individual o colaborativo.

3. Utilice técnicas de estudio:

    - Elija un entorno de estudio que resulte agradable y sin elementos que lo que distraigan

    - Estudie activamente, para esto lea en voz alta, tome notas, elabore esquemas o mapas conceptuales, realice resúmenes.

    - Reflexione sobre lo que va aprendiendo, para esto relacione lo nuevo con lo anterior o conocido, asegúrese de que entiende y es capaz de aplicar lo que está aprendiendo antes de pasar adelante.

4. Interactúe con los materiales de estudio en tres fases:

    - Fase de aproximación: Revise el objetivo de la unidad y después la acción o acciones a lograr; busque los materiales sugeridos para desarrollar la tarea, verifique cuál es el tiempo de que dispone.

    - Fase de lectura profunda: aproxímese al material a través de una lectura ligera, poniendo especial interés en los títulos y subtítulos. Trate de relacionar lo que va leyendo con conocimientos adquiridos previamente.

    - Fase de evaluación: Una vez realizada la lectura, intente realizar las actividades de auto-aprendizaje.

4. Requisitos Previos:

Tener conocimientos básicos de computación: los estudiantes matriculados deben conocer los procedimientos básicos para el manejo de PC y/o dispositivos móviles, aplicaciones de ofimática; así como uso de navegadores de páginas Web y el correo electrónico.

5. Actividades de auto-aprendizajes:

1. En los métodos recursivos se es necesario definir dos elementos, partes, casos. Mencione cuales son dichos elementos, partes, casos . En el código presentado en la **Introducción a la unidad didáctica** identifique dichos elementos o partes.

2. Entre los elementos que define los métodos o funciones recursivas son los casos bases. Que características tienen dentro de un método o función recursiva. Cuantos casos bases puede tener una función o método recursivo?.

3. Que pasaría si se llama o invoca a una función o método recursivo que no presenta ningún caso base en su implementación ?.

4. Entre los elementos que define los métodos o funciones recursivas son los casos recurrente. Que características tienen dentro de un método o función recursiva. Cuantos casos recurrentes puede tener una función o método recursivo?.

5. Que pasaría si se llama o invoca a una función o método recursivo que no presenta ningún caso recurrente en su implementación ?. Se podría considerar como un método recurrente.

6. Realice una tabla comparativa entre las ventajas y desventajas del métodos recursivos frente a métodos no recursivo.

6. Actividades de evaluación:

1.Declare e implemente una función o método recursivo que permita calcular el termino N de la sucesión de los números Fibonacci. Realice el árbol de traceo para N= 5

2.Declare e implemente una función o método recursivo que permita calcular el máximo común divisor entre dos números enteros A y B que son pasados por parámetros. Realice el árbol de traceo para A= 24 y B= 102

3.Declare e implemente una función o método recursivo que calcule la función de Ackermann para dos valores enteros positivos N y M que son pasados por parámetros. Realice el árbol de traceo para N=10 y M=15.

4. Declare e implemente una función o método recursivo que permita calcular el termino N de la secuencia de los números Catalan. Realice el árbol de traceo para N= 10.

7. Resumen:

Recursión es, en ciencias de computación, una forma de atajar y solventar problemas. De hecho, recursión es una de las ideas centrales de ciencia de computación. Resolver un problema mediante recursión significa que la solución depende de las soluciones de pequeñas instancias del mismo problema.

En el ejemplo "factorial" la implementación iterativa es probablemente más rápida en la práctica que la recursiva. Este resultado es lógico, pues las funciones iterativas no tienen que pagar el exceso de llamadas de funciones como en el caso de las funciones recursivas, y ese exceso es relativamente alto en muchos lenguajes de programación.

Hay otros tipos de problemas cuyas soluciones son inherentemente recursivas, porque estar al tanto del estado anterior. Un ejemplo es el DFS ; otros incluyen la función de Ackermann y el algoritmo divide y vencerás tales como Quicksort. Todos estos algoritmos pueden implementarse iterativamente con la ayuda de una pila, pero la necesidad del mismo, puede que anule las ventajas de la solución iterativa.

Otra posible razón para la utilización de un algoritmo iterativo en lugar de uno recursivo es el hecho de que en los lenguajes de programación modernos, el espacio de pila disponible para un hilo es, a menudo, mucho menos que el espacio disponible en el montículo, y los algoritmos recursivos suelen requerir más espacio de pila que los algoritmos iterativos.

8. Glosario de términos:

Consulte la bibliografía y defina el concepto del siguiente término:

Algoritmo recursivo:

Programación recursiva:

Recursión directa e indirecta:

9. Bibliografía:

Aprenda Java como si estuviera en primero. *Colectivo de autores*. Cap 3 epígrafes 3.1, 3.2, 3.3, 3.4.

Cómo programar en Java. *Deitel, Paul J. Y Harvey M. Deitel.* Cap 6 completo

10. Recursos educativos digitales RED:

11. **Próxima unidad:** Programación funcional. Ejercitación de los contenidos.