Recursividad

La recursividad es una técnica de programación importante. Se utiliza para realizar una llamada a una función desde la misma función. Como ejemplo útil se puede presentar el cálculo de números factoriales. Él factorial de 0 es, por definición, 1. Los factoriales de números mayores se calculan mediante la multiplicación de 1 * 2 * ..., incrementando el número de 1 en 1 hasta llegar al número para el que se está calculando el factorial.

La recursividad y la iteración (ejecución en bucle) están muy relacionadas, cualquier acción que pueda realizarse con la recursividad puede realizarse con iteración y viceversa. Normalmente, un cálculo determinado se prestará a una técnica u otra, sólo necesita elegir el enfoque más natural o con el que se sienta más cómodo.

Claramente, esta técnica puede constituir un modo de meterse en problemas. Es fácil crear una función recursiva que no llegue a devolver nunca un resultado definitivo y no pueda llegar a un punto de finalización. Este tipo de recursividad hace que el sistema ejecute lo que se conoce como bucle "infinito".

Casos:

- TRIVIALES: Determinar para qué valores de los parámetros (bajo qué condiciones), existe una solución directa y cuál es esa solución. Debe existir como mínimo 1 caso trivial, si no el algoritmo no parará de generar subproblemas.

- GENERALES: Determinar para qué valores de los parámetros la solución se obtiene por descomposición en subproblemas del mismo tipo. Establecer cómo se calculan las soluciones resueltas las llamadas recursivas correspondientes.
  Comprobar que las llamadas recursivas usan exactamente los parámetros definidos y que el tamaño del problema se reduce o tiende hacia los casos triviales.

Tipos.

Podemos distinguir dos tipos de recursividad:

Directa: Cuando un subprograma se llama a sí mismo una o más veces directamente. Indirecta: Cuando se definen una serie de subprogramas usándose unos a otros.

Características.

Un algoritmo recursivo consta de una parte recursiva, otra iterativa o no recursiva y un a 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.

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.

Propiedades de procedimientos recursivos

Debe existir criterio base para que este se llame a sí mismo.

Cada vez que el procedimiento se llame a si mismo debe estar más cerca del criterio base

Bibliografía:

Tenenbaum, Aaron. Estructura de Datos en C. México 1995.