Introducción

Un poco de historia

La noción de «algoritmos» es básica para la programación de las computadoras. Se la asoció con «logaritmo» en un principio. Su primera aparición fue en 1957, con el nombre de «algoritmo», cuyo significado era realizar cálculos aritméticos usando guarismos aravicos. Luego, los matemáticos encontraron el origen de esta palabra en un libro árabe. En 1950, esta palabra era asociada al algoritmo de Euclides.

Definición de algoritmo

Conjunto finito de reglas que dan una secuencia de operaciones para resolver un determinado problema.

Características

Finitud

Un algoritmo debe terminar siempre después de un determinado número de pasos.

Definibilidad

Cada paso de un algoritmo debe definirse de modo preciso.

Entrada

Debe tener cero o más entradas, dadas antes de empezar el algoritmo.

Salida

Debe tener una o más salidas, dependiendo de la entrada de datos.

Efectividad

Todas las operaciones realizadas por el algoritmo deben ser básicas, de tal manera que puedan ser resueltas por el hombre en un tiempo finito, y en forma exacta, utilizando lápiz y papel.

Para un mismo problema se pueden diseñar distintos algoritmos que lo resuelvan correctamente. Se debe determinar qué algoritmo utilizar.

Un algoritmo es mucho mejor cuantos menos recursos consuma, pero se deben tener en cuenta otros factores antes de establecer la conveniencia o no de éste. Por ejemplo, la facilidad para programarlo, la facilidad de entenderlo, y el criterio de eficiencia buscado: relación entre recursos consumidos y productos conseguidos. En general, es posible decir que los recursos que consume un algoritmo dependen de factores internos y externos.

Factores internos:

- Tamaño de los datos de entrada.

Factores externos:

- Procesador, memoria, etc. donde se ejecuta el programa.
- El lenguaje de programación y el compilador usado.
- La implementación que haga el programador del algoritmo, en particular de las estructuras de datos usadas.

Pero antes de hablar de la eficiencia propiamente dicha, es necesario recordar que el diseño de un algoritmo puede ser iterativo o recursivo.

Algoritmos recursivos

Concepto

Técnica de programación muy potente que puede ser usada en lugar de la iteración. Permite definir algoritmos de una forma simple.

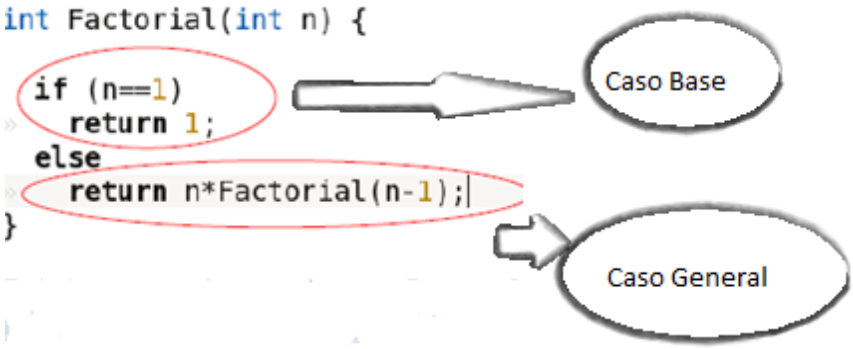
Una función es recursiva si, dentro de su código, se llama a sí misma.

Toda función recursiva tiene 2 componentes:

1. Caso base: contiene la condición de parada o fin de la recursividad.
2. Caso general: contiene el código recursivo.

Ejemplo

La función factorial:



1) factorial(4) := 4 * factorial(3) Se invoca a sí misma y crea una segunda variable, cuyo nombre es número y su valor es igual a 3.
2) factorial(3) := 3 * factorial(2) Se invoca a sí misma y crea una tercera variable, cuyo nombre es número y su valor es igual a 2.
3) factorial(2) := 2 * factorial(1) Se invoca a sí misma y crea una cuarta variable, cuyo nombre es número y su valor es igual a 1.
4) factorial(1) := 1 * factorial(0) Se invoca a sí misma y crea una quinta variable, cuyo nombre es número y su valor es igual a 0.
5) Como factorial(0) := 1, con éste valor se regresa a completar la invocación: factorial(1) := 1 * 1 , por lo que factorial(1) := 1.
6) Con factorial(1) := 1, se regresa a completar: factorial(2) := 2 * 1, por lo que factorial(2) := 2.
7) Con factorial(2) := 2, se regresa a completar : factorial(3) := 3 * 2, por lo que factorial(3) := 6.
8) Con factorial(3) := 6, se regresa a completar : factorial(4) := 4 * 6, por lo que factorial(4) := 24, y éste será el valor que la función factorial devolverá al módulo que la haya invocado, con un valor de parámetro local igual a 4 .

Tipos de recursividad

Directa

El módulo recursivo se llama a sí mismo.

Indirecta

Se tienen varios módulos que se llaman unos a otros, formando un ciclo.

Ejemplo

Funciones par e impar:

- N es par si n-1 es impar.
- N es impar si n-1 es par.

Lineal

Existe una única invocación recursiva.

Múltiple

Existe más de una invocación recursiva.

Pero, ¿qué sucede con la memoria?

La memoria del ordenador se divide (de manera lógica, no física) en cuatro segmentos:

1) Segmento de código

Parte de la memoria donde se guardan las instrucciones del programa en código de máquina.

2) Segmento de datos

Parte de la memoria destinada a almacenar las variables estáticas.

3) Montículo

Parte de la memoria destinada a las variables dinámicas.

4) Pila

Parte destinada a las variables locales y parámetros de la función que está siendo ejecutada.

Cuando se llama a una función en un algoritmo, en la memoria:

- Se reserva espacio en la pila para los parámetros de la función y sus variables locales.
- Se guarda en la pila la dirección de la línea de código desde donde se ha llamado a la función.
- Se almacenan los parámetros de la función y sus valores en la pila.
- Al terminar la función, se libera la memoria asignada en la pila y se vuelve a la instrucción actual.

En el caso de una función recursiva, cada llamada genera un nuevo ejemplar de la función con sus correspondientes objetos locales:

- La función se ejecutará normalmente hasta la llamada a sí misma. En ese momento, se crean, en la pila, nuevos parámetros y variables locales.
- El nuevo ejemplar de función comienza a ejecutarse.
- Se crean más copias hasta llegar a los casos bases, donde se resuelve directamente el valor, y se va liberando memoria hasta llegar a la primera llamada (última en cerrarse).

La medida de la eficacia en un algoritmo iterativo o recursivo viene dada por:

- Tiempo de ejecución.
- Espacio de memoria ocupado por el algoritmo.