Algoritmos y Programación II

09−Abril−1997

Recursividad

Un objeto es recursivo si está parcial o totalmente definido en términos de sí mismo. Permite que algo que sea infinito pueda ser definido de forma finita. Un programa es recursivo si se puede llamar a sí mismo.

Toda definición recursiva tiene dos partes:

Base: Se definen los elementos del conjuntos que servirán para crear el resto del conjunto con la ayuda de:

Reglas de recurrencia: Reglas que permiten crear un elemento a partir de elementos ya creados.

Ejemplo:

1. Algoritmo para calcular el factorial de un numero entero y positivo. Si $n \geq 0$, se define,

$n! = n(n-1)(n-2)...3 \cdot 2 \cdot 1$

$n! = n \cdot (n-1)!$

Necesitamos una base : si cumple una condición deberemos acabar la ejecución para evitar bucles infinitos.

(a) 0! = 1

(b) si n>0 aleshores n! = n*(n−1)!

(a) es la base y (b) la regla de recurrencia.

El algoritmo recursivo para calcular el factorial es:

funcio fact(n:enter):enter

inici

si n=0 **aleshores**

fact<−1 {(a)}

sino

fact<−n*fact(n−1) {(b)}

fisi

fi

Este algoritmo es recursivo, no iterativo. Cuando se usen bucles *for* o *while* no se deberá implementar

recursividad.

2. Sucesión de Fibonacci: 1,1,2,3,5,8,... an=an−1+an−2

(a) Definición recursiva:

Si n>1 fib(n) = fib (n−1) + fib (n−2)

si n = 0 o n=1 fib (0) = fib (1) = 1

(b) Algoritmos:

funcio Fibonacci (n: enter):enter;

inici

fib<−1

if n<>0 i n<>1 **aleshores**

fib = Fibonacci (n−1) + Fibonacci (n−2)

fisi

Fibonacci = fib

fi

Para una n=4, habremos llamado 9 veces a la función Fibonacci.