25-1-2022

Recursión/inducción

Tarea: 2

Materia: Seminario de estructura de datos 1

Sección: D13.

Código: 216584703

Carrera: Ingeniería en computación.

Nombre alumno: Padilla Pérez Jorge Daray

Nombre profesor: Julio Esteban Valdes Lopez

Índice general

Que es la recursividad 2
¿Cómo se calcula un factorial? **3**
¿Por qué escribir programas recursivos? 3
Ventajas y desventajas 4
Condiciones necesarias 4
Tipos de recursión **4**
Recursión lineal 4
Recursión lineal no final 4
Recursión lineal final 4
Recursión múltiple 4
Recursión mutua 4
Ejemplos de recursividad **5**
La sucesión de Fibonacci 5
Máximo común divisor 5
Suma de factores 5
La inducción matemática **7**
¿Como se resuelve? 7
Ejemplos de inducción matemática **8**
Codigo ejemplo de recursividad **9-10**
Referencias bibliográficas **11**

¿Qué es la recursividad?

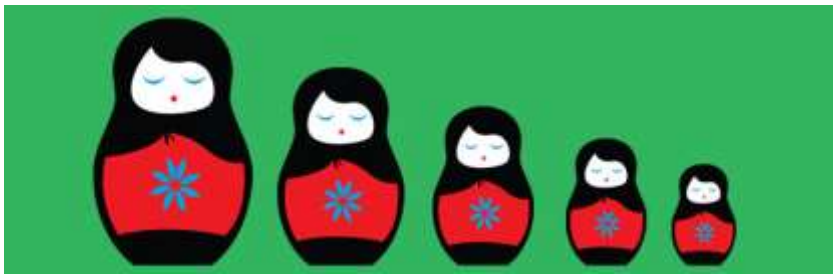
La recursividad es un concepto fundamental en matemáticas y en computación.

Los términos **recurrencia, recursión** o **recursividad** hacen referencia a una técnica de definición de conceptos (o de diseño de procesos) en la que el concepto definido (o el proceso diseñado) es usado en la propia definición(diseño).

La recursividad, también llamada recursión o recurrencia, es la forma en la cual se especifica un proceso basado en su propia definición. O sea, si se tiene un problema de tamaño N, este puede ser dividido en instancias más pequeñas que N del mismo problema y conociendo la solución de las instancias más simples, se puede aplicar inducción a partir de estas asumiendo que quedan resueltas.

• Es una alternativa diferente para implementar estructuras de repetición (ciclos). Los módulos se hacen llamadas recursivas.

• Se puede usar en toda situación en la cual la solución pueda ser expresada como una secuencia de movimientos, pasos o transformaciones gobernadas por un conjunto de reglas no ambiguas.



Vulgarmente se le conoce como la Matrushka, La Matrushka es una artesanía tradicional rusa. Es una muñeca de madera que contiene otra muñeca más pequeña dentro de sí. Esta muñeca, también contiene otra muñeca dentro. Y así, una dentro de otra.

¿Por qué escribir programas recursivos?

• Son más cercanos a la descripción matemática.

• Generalmente más fáciles de analizar

• Se adaptan mejor a las estructuras de datos recursivas.

• Los algoritmos recursivos ofrecen soluciones estructuradas,

modulares y elegantemente simples.

Ventajas y desventajas

Factible de utilizar recursividad:

• Para simplificar el código.

• Cuando la estructura de datos es recursiva

ejemplo : árboles.

No factible utilizar recursividad:

• Cuando los métodos usen arreglos largos.

• Cuando el método cambia de manera

impredecible de campos.

• Cuando las iteraciones sean la mejor opción.

Condiciones necesarias

Para dar solución a un problema a partir del empleo de la recursividad, es necesario asegurar que cada llamada recursiva debe estar definida sobre un problema menos complejo que el que dio lugar a la llamada, y que debe existir al menos un caso base al que se garantice la llegada del problema en un momento determinado. Si no se cumpliera esta condición se generaría una secuencia infinita de llamadas y el algoritmo que intenta dar solución al problema no terminaría nunca.

Tipos de recursión

Como regla básica, para que un problema pueda resolverse utilizando recursividad, el problema debe poder definirse recursivamente y, segundo, el problema debe incluir una Metodología y tecnología de la programación I condición de terminación porque, en otro caso, la ejecución continuaría indefinidamente. Cuando la condición de terminación es cierta la función no vuelve a llamarse a si misma. Pueden distinguirse distintos tipos de llamada recursivas dependiendo del número de funciones involucradas y de cómo se genera el valor final. A continuación, veremos cuáles son.

Recursión lineal

En la recursión lineal cada llamada recursiva genera, como mucho, otra llamada recursiva. Se pueden distinguir dos tipos de recursión lineal atendiendo a cómo se genera resultado.

Recursión lineal no final

En la recursión lineal no final el resultado de la llamada recursiva se combina en una expresión para dar lugar al resultado de la función que llama. El ejemplo típico de recursión lineal no final es cálculo del factorial de un número (n! = n * (n-1) * ...* 2 * 1). Dado que el factorial de un número n es igual al producto de n por el factorial de n-1, lo más natural es efectuar una implementación recursiva de la función factorial. Veamos una implementación de este cálculo:

Recursión lineal final

En la recursión lineal final el resultado que es devuelto es el resultado de ejecución de la última llamada recursiva. Un ejemplo de este cálculo es el máximo común divisor, que puede hallarse a partir de la fórmula:

$$mcd(n,m) = \begin{cases} n & n = m \\ mcd(n-m,m) & n > m \\ mcd(n,m-n) & n < m \end{cases}$$

Recursión múltiple

Alguna llamada recursiva puede generar más de una llamada a la función. Uno de los centros más típicos son los números de Fibonacci, números que reciben el nombre del matemático italiano que los descubrió. Estos números se calculan mediante la fórmula:

$$F(n) = \begin{cases} 1, & \text{si } n = 1; \\ 1, & \text{si } n = 2; \\ F(n-1) + F(n-2) & \text{si } n > 2; \end{cases}$$

Recursión mutua

Implica más de una función que se llaman mutuamente. Un ejemplo es el determinar si un número es par o impar mediante dos funciones:

Ejemplos de recursividad

La sucesión de Fibonacci es conocida desde hace miles de años, pero fue Fibonacci (Leonardo de Pisa) quien la dio a conocer al utilizarla para resolver un problema.

El **primer** y **segundo** término de la sucesión son

$$a_0 = 0$$

$$a_1 = 1$$

Los siguientes términos se obtienen sumando los dos términos que les preceden:

El **tercer término** de la sucesión es

$$a_2 = a_0 + a_1 = \\ = 0 + 1 = 1$$

El **cuarto término** es

$$a_3 = a_1 + a_2 = \\ = 1 + 1 = 2$$

El **quinto término** es

$$a_4 = a_2 + a_3 = \\ = 1 + 2 = 3$$

El **sexto término** es

$$a_5 = a_3 + a_4 = \\ = 2 + 3 = 5$$

El (n+1)(n+1)**-ésimo término** es

$$a_n = a_{n-2} + a_{n-1}$$

Revisar practica 1 para el ejemplo en código

Máximo común divisor

Vamos a calcular el máximo común divisor de 12 y 18.

Puesto que el número que buscamos tiene que dividir a 12 y a 18, no puede ser mayor que 12.

En la siguiente tabla escribimos los candidatos:

Puesto que el máximo común divisor debe dividir a los dos números, las únicas posibilidades son: 1, 2, 3 y 6.

Como tiene que ser el más grande posible, el MCD es 6.

$$M.C.D.(12,18) = 6$$

	Divisor de **12**?	Divisor de **18**?
1	Sí	Sí
2	Sí	Sí
3	Sí	Sí
4	Sí	No
5	No	No
6	→ Sí	→Sí
7	No	No
8	No	No
9	No	Sí
10	No	No
11	No	No
12	Sí	No

Suma de vectores

Primero se crean vectores al azar de entre 1 a 4 dígitos.

En seguida con recursividad vas sumando el anterior de manera que queda así:

Vector [n+1] + vector (vector, n);

3	Nº 1	12	0,8634156
4	Nº 2	3	0,21355013
5	Nº 3	10	0,72431138
6	Nº 4	13	0,98197894
7	Nº 5	1	0,06196648
8	Nº 6	10	0,77306183
9	Nº 7	9	0,67559748
10	Nº 8	6	0,44831591
11	Nº 9	9	0,70210203
12	Nº 10	2	0,17882113
13			
14	Comprobación:	75	

La inducción matemática

La inducción matemática es un método que se usa para demostrar afirmaciones acerca de todos los números naturales o de todos los naturales a partir de alguno. Consideremos la afirmación:

Todo número mayor o igual a 7 es suma de un múltiplo de 3 y un múltiplo de 4.

¿Como podemos saber si la afirmación es cierta? Podemos ver que

$7=1\cdot3+1\cdot4$, $8=0\cdot3+2\cdot4$, $9=3\cdot3+0\cdot4$, $10=2\cdot3+1\cdot4$, $11=1\cdot3+2\cdot4$, $137=23\cdot3+17\cdot4$, pero ver que es cierto en un montón de casos particulares no demuestra que sea siempre cierta, no importa cuantos casos probemos, siempre podría fallar en el siguiente!

¿Como se resuelve?

¿Como podemos saber que siempre es cierta, si no podemos comprobarlo en cada caso, ya que nos tomaría una eternidad? La inducción es un tipo de razonamiento lógico que permite hacer demostraciones para una infinidad de casos en un tiempo finito.

Una demostración por inducción tiene dos pasos: 1. (La base de la inducción) Mostrar que la afirmación es cierta en el primer caso. 2. (El paso de inducción) Suponer que la afirmación es cierta en un caso, y mostrar que debe ser cierta en el siguiente caso.

Si podemos demostrar que una afirmación es cierta en el primer caso y podemos demostrar que siempre que la afirmación sea cierta en un caso también debe ser cierta en el caso siguiente, entonces también debe ser cierta en el caso 2, y por lo tanto debe ser cierta en el caso 3, y también debes ser cierta en el caso 4, etc., ¡así que no puede haber ningún caso en que falle!.

Ejemplos de inducción matemática

• Todo número mayor o igual a 7 es suma de un múltiplo de 3 y un múltiplo de 4.

Demostración por inducción:

Paso 1. (Base de inducción) Demostramos que la afirmación es cierta en el primer caso. 7 si es suma de un múltiplo de 3 y un múltiplo de 4, ya que $7=1\cdot3+1\cdot4$

Paso 2. (Paso de inducción) Suponemos que la afirmación es cierta para un n y debemos mostrar que entonces es cierta para n+1. Hipóstasis de inducción: n es suma de un múltiplo de 3 y un múltiplo de 4 Por demostrar: n+1 es suma de un múltiplo de 3 y un múltiplo de 4. Demostración Si $n=r\cdot3+s\cdot4$ entonces $n+1=r\cdot3+s\cdot4+1$ y necesitamos ver si esto es la suma un múltiplo de 3 y uno de 4. En efecto, $r\cdot3+s\cdot4+1 = (r-1)\cdot3+(s+1)\cdot4$ si $r>0$, y $r\cdot3+s\cdot4+1 = (r+3)\cdot3+(s2)\cdot4$ si $r=0$ (como $n>7$, si $r=0$ entonces $s\geq2$). Así que si la afirmación es cierta para un numero n entonces también es cierta para el numero n+1. □

• Demuestra que para todo número natural n se tiene $1+2+3+...+n = n(n+1)/2$.

Paso 1 Base de inducción: $1 = 1(1+1)/2$

Paso 2. Hipótesis de inducción: $1+2+3+...+n = n(n+1)/2$

Por demostrar: $1+2+3+...+n+1 = (n+1)(n+2)/2$

Demostración. $1+2+3+...+n+1 = (1+2+3+...+n)+(n+1) = n(n+1)/2 + (n+1) = =an(n+1)+2(n+1)/2 = (n+2)(n+1)/2$ que es lo que queríamos demostrar.

• Demuestra que $n! > 2n$ para toda $n \geq 4$.

Paso 1 Base de inducción: 4!=24 y 24=16, así que 4!>24

Paso 2. Hipótesis de inducción: $n! > 2n$

Por demostrar: $(n+1)! > 2n+1$

Demostración. Si $n! > 2n$

entonces $(n+1)! = n!(n+1) > 2n (n+1)$ Como $n \geq 4$, $n+1 \geq 5$, así que $2 n (n+1)\geq2n (5)>2n (2)=2n+1$. Así que $(n+1)! > 2n+1$

| Código fuente ejemplo

Código ejemplo recursividad

```
#include <iostream>
using namespace std;

long fibonacci(int n){
if (0 == n || 1 == n) {
return n;
} else {
return (fibonacci(n-1) + fibonacci(n-2));
}
}

int main()
{
cout<<"\n\t\tFIBONACCI \n\n";

int i, numero ;
do
{
cout<<"Ingrese un numero entero y positivo: ";
cin>>numero;

} while(numero < 0);

cout<<"\nLa serie es: \n\n\t";

for(i=0; i<numero; i++)
{
if(fibonacci(i) != 0)
```

```
cout<< ", ";


cout<< fibonacci(i);
}
cout<<endl;
system("pause");

return 0;
}
```

Referencias bibliográficas

N.A. (2012). Recursividad. 4 de marzo de 2021, de uv Sitio web: https://www.uv.mx/personal/ocastillo/files/2011/04/Recursividad.pdf

Chávez Agüero. (2020). Recursividad . 4 de marzo de 2021, de ecured Sitio web: https://www.ecured.cu/Recursividad

José Manuel Aragón . (2018). ¿Qué es la recursividad?. 4 de marzo de 2021, de campusmvp Sitio web: https://www.campusmvp.es/recursos/post/Que-es-la-recursividad-o-recursion-Un-ejemplo-con-JavaScript.aspx#:~:text=As%C3%AD%2C%20por%20ejemplo%2C%20el%20factorial,5%20ser%C3%ADa%3A%205x4x3x2x1%20%3D%20120.&text=Aqu%C3%AD%20lo%20que%20se%20hace,que%20hay%20en%20el%20condicional).

J.M.Gimeno y J.L. González. (2008). Recursividad. 4 de marzo de 2021, de ocw Sitio web: http://ocw.udl.cat/enginyeria-i-arquitectura/programacio-2/continguts-1/2-recursividad.pdf

N.A. (2016). N6 inducción matemática. 4 de marzo de 2021, de matem Sitio web: https://www.matem.unam.mx/~max/AS1/N6.pdf

N.A. (2017). Inducción . 4 de marzo de 2021, de cimat Sitio web: https://www.cimat.mx/especialidad.seg/anterior/documentos/celaya/induccion.pdf

Courant R, H. Robbins. ¿Qué son las Matemáticas? FCE, México 2002.