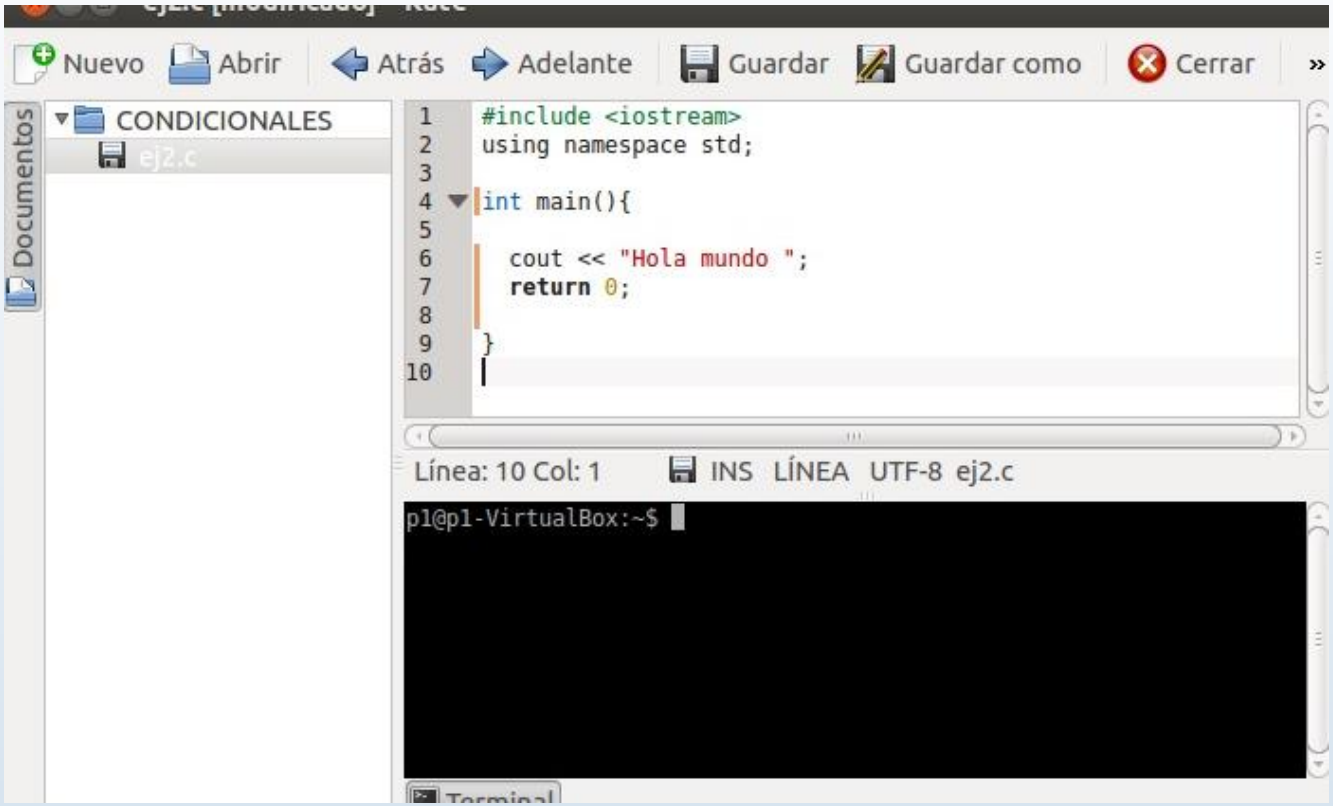


Programación 1

Tema 5. Recursividad

Grado en Ingeniería Informática

Objetivos / Competencias

1. Entender el concepto de recursividad
2. Saber diseñar algoritmos recursivos sencillos e implementarlos en lenguaje C
3. Comprender la ejecución de un módulo recursivo mediante la realización de trazas

Índice

Definición

- Un módulo es recursivo cuando entre la lista de instrucciones que lo forman, se encuentra una llamada a sí mismo, directa o indirectamente.

- Hay muchas funciones matemáticas que se definen de forma natural de manera recursiva. Por ejemplo:

  - Factorial de un número *n*: El factorial de un número *n* es el número *n* multiplicado por el factorial de *n-1*.

    $$factorial(n) = n * factorial(n-1)$$

  - Potencia de dos números: $x^n = x * x^{n-1}$

Esquema básico de un módulo recursivo

- Uno o más casos base
  - No hay llamadas recursivas en ellos. Especifican la "condición de terminación" o "condición de paro" de la recursión.

- Uno o más casos generales o recursivos
  - Incluye una o más llamadas al módulo. Estas llamadas recursivas deben resolver versiones "más pequeñas" de la tarea inicial que tiene que resolver el módulo. Es decir, tiene que haber un progreso o tendencia al caso base.

Ejemplo del factorial

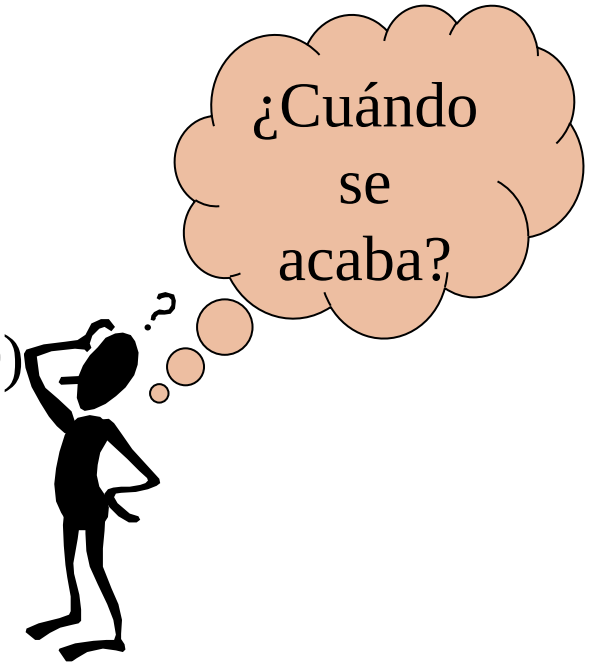
factorial (n)= n* factorial(n-1)

factorial (3) = 3*factorial(2)

= 2*factorial(1)

= 1*factorial(0)

= 0*…



¡ESTO TIENE MALA PINTA!

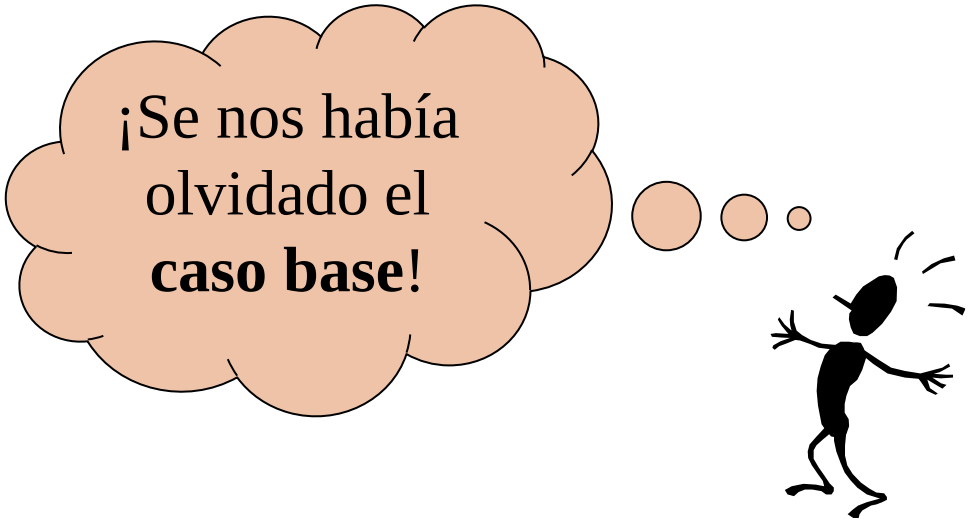
Ejemplo del factorial

Si n es igual a 0 Entonces

factorial = 1

Si no

factorial = n* factorial de (n-1)



Ejemplo del factorial

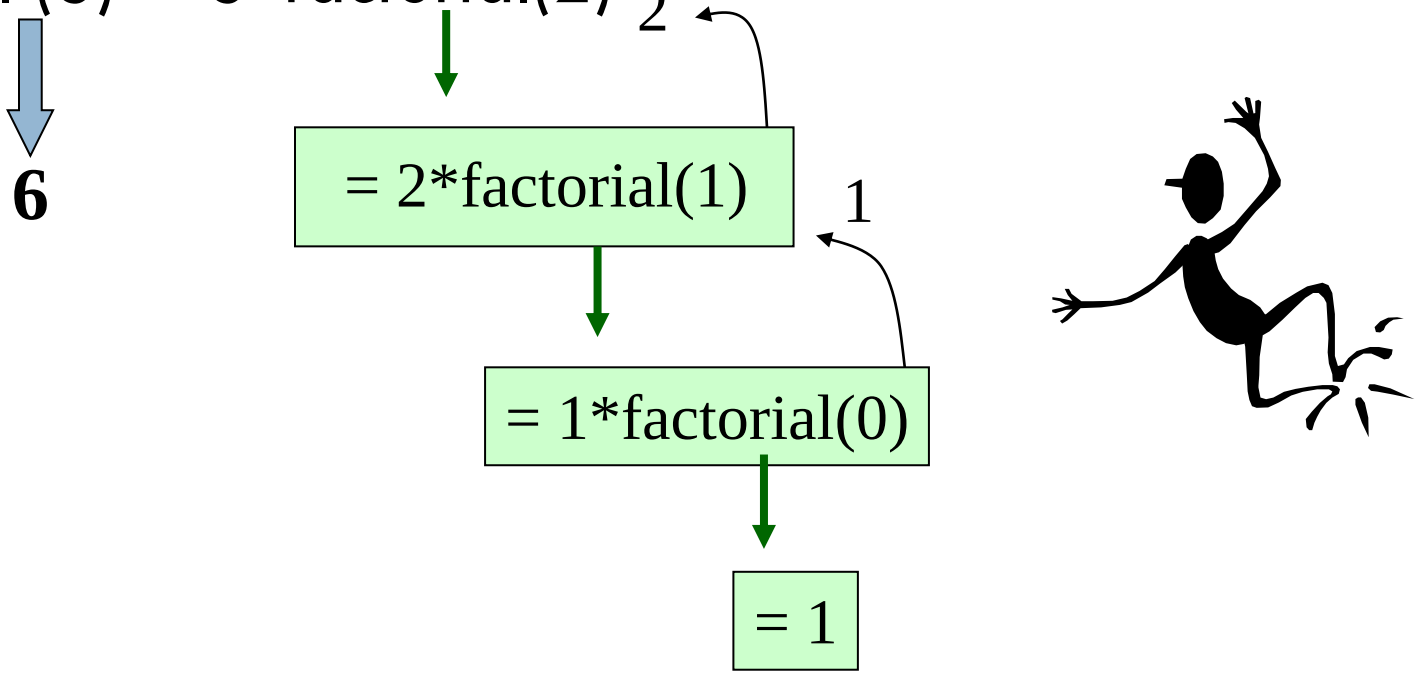
Si (n>0) Entonces

factorial (n)= n* factorial (n-1)

Si no

factorial (n)= 1

Factorial (3) = 3*factorial(2)



Codificación en C


```
#include <iostream>
using namespace std;
int factorial (int n)
{
   int res;
  if (n>0) // caso recursivo
      res = n * factorial(n-1);
   else // caso base
      res = 1;
   return res;
}

main()
{
    int num;
    cout << "Introduce un número";
    cin >> num;
    cout << factorial(num);
}
```

Características

- Idónea para la resolución de aquellos problemas que pueden definirse de modo natural en términos recursivos
- Tiene su equivalente iterativo
- Necesitan mayor cantidad de memoria para su ejecución
- Son más lentos en su ejecución

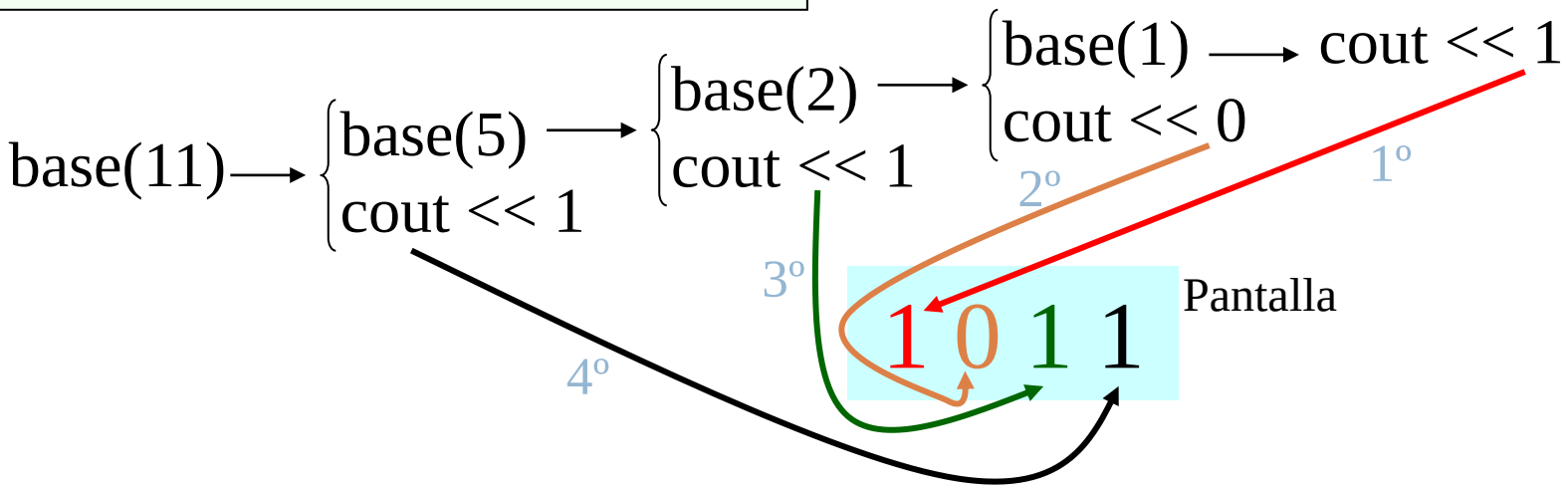
RECURSION INFINITA

```
void escribe (int n){
        escribe(n/10);
        cout << n%10 << endl;
}
```

Traza


```
void base (int n){
      if (n<2) // caso base
          cout << n;
      else{   // caso recursivo
           base (n/2);
           cout << n%2;
      }
}
```

```
main()
{
     base(11);
}
```



Ejemplo

Dado el siguiente módulo:

```
void recursivo (int num)
{
      if (num != 0){ // caso recursivo
            recursivo(num/2);
            cout << num % 2;
      }
}
```

1. ¿Cuál es la salida que se obtiene si se le llama de la siguiente forma: `recursivo(16)`?

   A) 00001

   B) 11111

   C) 10000

   D) 00100

   E) ninguna de las anteriores

2. ¿Cuál es el caso base?

¿Qué hace este código?


```
void alreves(char l){
    if (l == '.')  // caso base
      cout << endl;
     else{  // caso recursivo
         cin >> l;
         alreves(l);
         cout << l;
     }
 }
```

```
main(){
     char letra;

     cout << "Introduce una frase terminada en punto :";
     cin >> letra;
     alreves(letra);
     cout << letra;
}
```

Ejercicios

1. Diseñar un módulo recursivo que para un número natural *n* muestre por pantalla la serie creciente de números naturales del 1 al *n*, es decir, 1 2 3… n.
2. Diseñar un módulo recursivo que para un número natural *n* devuelva la suma de los cuadrados de los números del 1 hasta el n. Por ejemplo, para n=4, el módulo debe devolver 30 ya que $1^2 + 2^2 + 3^2 + 4^2 = 30$.
3. Diseñar un módulo que, dado un número natural, muestre por pantalla el número formado por los mismos dígitos en sentido contrario. Por ejemplo: para el número 2089 debe mostrar 9802.
4. Diseñar un módulo que reciba un número en sistema decimal y muestre en pantalla su equivalente en binario. Por ejemplo, para el número 12, debe mostrar en pantalla 1100.
5. Implementa una función recursiva que devuelva el número de dígitos impares de un número. Ejemplo: rec(321)=2, rec(28)=0.